2006

One-dimensional computer modeling of electrical conductivity through methane and synthesis gas flames

Jonathan Patrick Lilly
West Virginia University

One-Dimensional Computer Modeling of Electrical Conductivity through Methane and Synthesis Gas Flames

Jonathan Patrick Lilly
BSEE, BSCpE, West Virginia University

Thesis
Submitted to
The College of Engineering and Mineral Resources
at
WEST VIRGINIA UNIVERSITY
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Examining Committee Members:

Dr. Roy S. Nutter Jr. (Committee Chair)
Dr. Mark A. Jerabek
Dr. Powsiri Klinkhachorn

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2006

Abstract

One-Dimensional Computer Modeling of Electrical Conductivity through Methane and Synthesis Gas Flames

Jonathan Patrick Lilly

The purpose of this work has been to produce and test simulation software to accurately model the effects of charged species and applied electric fields on the current conducted through methane and methane-doped synthesis gas flames. Comparison with existing experimental data generated at NETL has been conducted on a variety of published reaction mechanisms.

Acknowledgment

The research for this thesis was conducted as part of the continuing work on the Combustion Control and Diagnostic Sensor (CCADS) project at the National Energy Technology Laboratory (NETL) in Morgantown, West Virginia. The position was made available as under the Department of Energy's (DOE), Oak Ridge Associated Universities (ORAU) through the Oak Ridge Institute for Science and Education (ORISE).

Special thanks to David Huckaby, Benjamin Chorpening, Jimmy Thornton, and Clark Robinson who made my experience at NETL quite imformative and enjoyable. Also, to Dr. Roy S. Nutter, who chose me for this research and advised me throughout my graduate training.

Table of Contents

Keywords

Adaptive Grid
Arrhenius Function
Cantera
Charged Species
Charged Species Interaction
Chemical Kinetics
Chemical Transport Properies
Combustion
Combustion Control and Diagnostics Sensor (CCADS)
Conduction Current Density
Conductivity
Current Density
Diffusion
Diffusion Flux
Diffusivity
Drift
Dynamic Pressure Oscillation
Electric Field
Electric Potential
Electrostatic Drift Flux
Energy Conservation
Enthalpy
Entropy
Equivalence Ratio
Flame Ionization
Flashback
Flat Flame Burner
Homogeneous Reaction
Ionization Current
In-situ Combustion State Monitoring
Lean Blow-off
Lennard-Jones Collision Diameter
Lennard-Jones Well Depth
Lindemann Falloff Reaction
Mass Conservation
Mass Flux
Methane
Molecular Transport
NASA Thermodynamic Polynomial
No_x
One-Dimensional Computer Model
Premixed Combustion
Python
Reaction Rate Coefficient
Rotational Relaxation Collision Number
Saturation Current
Species Conservation
Specific Heat
Synthesis Gas
Thermal Conduction
Third-Body Collision Efficiencies
Three Body Reaction
Turbine Engine
Viscosity
XML

List of Symbols

A	-	pre-exponential coefficient of a reaction rate
a_i	-	one of the seven coefficients for an "old" NASA thermodynamic polynomial
c_p	-	constant-pressure specific heat
c_{p0}	-	reference constant-pressure specific heat
$D_{m,i}$	-	binary diffusion of species i in a mix
e	-	electron charge
E	-	activation energy for a chemical reaction
E_x	-	electric field at location x
H_0	-	reference enthalpy
h_i	-	enthalpy of species i
J_i	-	mass flux of species i
k_0	-	reaction rate as pressure approaches zero
k_∞	-	reaction rate as pressure approaches infinite
k_f	-	forward chemical reaction rate
n	-	temperature exponent for a reaction rate
N_A	-	Avogadro's number
P_r	-	non-dimensional reduced pressure
q	-	thermal conduction
S_0	-	reference entropy
T	-	Temperature
u	-	Velocity
W_i	-	molecular weight of species i
W_m	-	average molecular weight in a mix
X_i	-	mole fraction of species i
Y_i	-	mass fraction of species i
z_i	-	number of extra or missing electrons in a molecule of species i
ε_0	-	permittivity of free space
μ_i	-	mobility of species i
ν_{ij}	-	net stoichiometric coefficient for species k in reaction i
ρ	-	Density
Φ	-	electric potential
$\dot{\Omega}_j$	-	rate of progress of reaction j
$\dot{\omega}_i$	-	net production rate of species i

List of Figures

List of Tables

List of Equations

Chapter One: Introduction

1.1 Summary of CCADS

The Combustion Control and Diagnostics Sensor (CCADS) project is a sensor project meant to assist in turbine engines running with lean premixed air and fuel under conditions leading to low NO_x, low unburnt hydrocarbon, and high efficiency (Thornton et al., 2004). This goal is accomplished by allowing for increased stability of combustion in turbines for premixed air and fuel approaching the lean blow-off limit through the use of in-situ combustion state monitoring. This sensor is currently being developed by the U.S. Department of Energy's National Energy Technology Laboratory in Morgantown, West Virginia and Woodward Industrial Controls as part of a Cooperative Research and Development Agreement (CCRDA).

CCADS works by inferring flame characteristics based on the movement and generation of charged species within a flames combustion zone. The system makes use of a set of two electrodes, the guard electrode and the sense electrode. These electrodes are electrically isolated and side by side on the center body of the premixing nozzle (Thornton et al., 2004). The sense electrode is located closer to the inlet than the guard electrode, as illustrated in Figure 1-1.

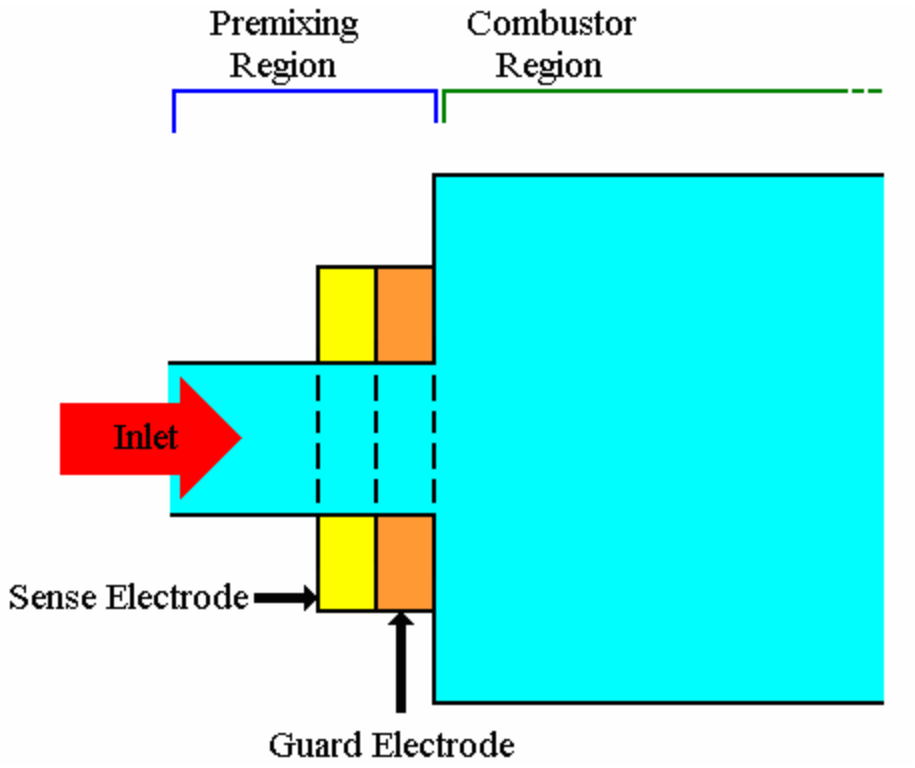

Figure 1-1: Basic CCADS Cross-Section

During operation, an equal electrical potential is applied to both the guard and the sense electrodes. Due to the electrode arrangement, under normal combustion conditions ionization current will be generated in the guard electrode only because the negative ions in the flame are preferentially attracted to the guard electrode when the flame is in the combustor region. If the flame enters the premixing region, however, ionization current will be detected through the sense electrode. This effect allows the sensor to detect flame flashback into the mixing chamber, a condition with the potential to cause significant hardware damage due to hot spots in the incomplete gas mixture. The response time of the sensor is fast enough to detect incipient flashback, allowing for corrective action to be taken (Thornton et al., 2004).

In addition to flashback, the CCADS sensor is capable of detecting incipient lean blow-off, a condition in which the flame goes out due to low fuel content in the premixed

combustion gas. Insipient lean blow-out is detected by detecting precursor events in which the flame momentarily extinguishes and reignites. These events can be detected because ionization current is only produced in the guard electrode when a flame is actively burning (Thornton et al., 2004).

Another feature which CCADS can monitor is dynamic pressure oscillations within a combustor. Dynamic pressure oscillations can lead to hardware failure due to heat transfer and vibration. Most modern turbine engines make use of a dynamic pressure transducer to detect pressure oscillations, but it has been shown that a correlation exists between the signal produced by a dynamic pressure transducer and the frequency of the ionization current in the CCADS guard electrode (Benson et al., 2003).

Finally, in addition to detecting the abnormal combustion conditions already discussed, CCADS may be used to provide a reasonably accurate measurement of the equivalence ratio of the premixed fuel under certain operating conditions. A link between current generated through the CCADS electrodes and equivalence ratia has been found experimentally, but there has not yet been success in calculating the connection quantitatively. However, the measurement of equivalence ratio may be incorrect under certain operating conditions such as in the presence of dynamic pressure oscillations and factors such as flame location and combustion temperature may further complicate the estimates.

Equivalence ratio is a ratio of the air and fuel in a combustion mixture in relation to a perfect mixture. An equivalence ratio of exactly one indicates that a mixture contains exactly enough air to combust all fuel in the mixture. This perfect mix is known as a stoichiometric mixture. Mixtures with an equivalence ratio greater than one are

called “rich” mixtures and do not contain enough air to combust all of the fuel. Mixtures with equivalence ratios less than one are called “lean” mixtures and contain more than enough air for full combustion. CCADS may be used to measure the equivalence ratio in lean mixtures because the average current detected by the sensor’s guard electrode increases as a linear function of increasing equivalence ratio in lean burning flames (Thornton et al., 2004).

1.2 Modeling Objectives

As part of the CCADS project, a one-dimensional computer model of flame ionization is being developed. The goal of this model is to predict ionization currents at various equivalence ratios in premixed air fuel mixtures. The mixtures used in this research include methane or syngas and air, but the model should be capable of working with other fuel types if a suitable reaction mechanism model is built. The real-world representation of the target for this model is a flat flame burner like the one illustrated in Section 3.1.1.

Combustion is a chemical kinetic process in which large numbers of chemical species are generated, destroyed, and diffuse within a combustion zone. In modeling of combustion it simply is not possible to model every molecule exactly because of the large amount of computational resources that would be required for an ideal model, and as a result some approximations are necessary. An almost limitless number of chemical species may be produced within a real flame, but a model cannot represent an unlimited number of species. As a result, a likely subset of reactions must be used to model flame chemistry. For these chemical subsets, various published reaction mechanisms were used. A second approximation is the use of grids in which representations of species

concentrations, temperatures, velocity and other flame properties are stored for various known positions, rather than an exact map of all molecules within the flame. Also, the interactions of species within the flame must be modeled based on real-world physics. However, in order to decrease computation time, it is common practice to ignore certain effects that do not play a significant role in a given application.

Modeling software is readily available which produces reasonably accurate representations of chemical kinetics within flames, but the available modeling software lacks the ability to accurately model the interactions of charged particles and applied electric fields within a combustion zone. The reason for this neglected functionality is that for most applications the interactions of charged species do not play a significant role. The charged species within a methane flame account for an almost insignificant proportion of the total mass within the flame, and in synthesis gas combustion, charges species are almost nonexistent. For CCADS, however, these charged species interactions are a vital interest.

1.3 Brief Comparison with Semiconductor Electronics

Although the goal of this research is to model charged particle movement within a combustion zone, there are some useful similarities with the field of semiconductor electronics. For example, the conduction current density given in Equation 1-1 for electrons and Equation 1-2 for holes (Streetman et al., 2000) follows the same form as the mass flux equation used in this modeling software as shown in Equation 2-5. In semiconductors and in combustion gas, current is produced by diffusion of charged particles and drift within applied electric fields.

$$J_n(x) = q\mu_n n(x)E(x) + qD_n \frac{dn(x)}{dx} \tag{1-1}$$

$$J_p(x) = q\mu_p p(x)E(x) - qD_p \frac{dp(x)}{dx} \tag{1-2}$$

As can also be seen in this research, ionization current reaches saturation once a high enough field is applied. Once saturation is reached in a flame, the current no longer increases with a higher applied field. In semiconductors, this effect is a result of reaching maximum drift velocity. A formula for drift velocity is shown in Equation 1-3 (Streetman et al., 2000).

$$v_d \cong \frac{\mu E}{1 + \mu E / v_s} \tag{1-3}$$

1.4 Statement of Problem

The purpose of this research was to test and provide input for the modification and extension of a one-dimensional reaction simulation code to incorporate the effects of charged species and applied electric fields. This modification and extension included automation of the code to perform multiple similar simulations, processing of simulation outputs in order to calculate currents within the simulated combustion flames, and provide verification by comparison with existing experimental data. Although this model may be applied to many types of fuels with the use of appropriate reaction mechanisms, the scope of simulation and experimentation was limited to methane and synthesis gas combustion. Synthesis gas is a mixture of CO and H2 derived as an alternative fuel from cooking coal, but in this work, the trace hydrocarbons in synthesis gas are ignored.

Chapter Two: Explanation of Model

2.1 Cantera

The goal of this project was to produce a one-dimensional model of premixed flames and apply the model to simulations with varying fuel mixes and flow rates. This model includes production, transport, and destruction of charged species and the effects of an applied electric field. Much work has already been accomplished in the area of simulation of chemical kinetics problems, including the development of a variety of programs meant to simulate chemical kinetics and transport. However, these programs generally ignore the effects of electrically charged species and applied electric fields within the reaction zone because charged species make up a very small proportion of the total mass of the system and in general practice an electric field is not applied across combustion zones. However, it is possible to use an existing program as a starting point for this model, but the original program must be modified in order to incorporate the effects of charged species and applied electric fields.

As a starting point for this model, a program called Cantera was chosen for use. Cantera is a set of chemical kinetic libraries written and maintained by Dr. David Goodwin, a professor of Mechanical Engineering at the California Institute of Technology (Goodwin, 2005, *Object-Oriented Software for Reactive Flows*). Cantera was written in order to accomplish many of the same tasks as the well known and widely used Chemkin program, which was developed at Sandia National Laboratories but is now maintained and distributed by Reaction Design. Cantera also contains additional functionality not available with Chemkin such as reactor networks and a Matlab interface. However, unlike Chemkin, Cantera is an open source program and, as a result,

it can be freely downloaded and modified. Cantera is available in source and binary distributions on its Sourceforge project webpage (Goodwin, 2005, *Sourceforge.net: Project Info – Cantera)*. In addition, Cantera includes interfaces for C++, Python, FORTRAN, and Matlab. For the simulation performed for this project, the Python interface was used. Python interpreters can be freely downloaded online (Python Software Foundation, 2005).

The following sections will discuss some of the inner workings of the Cantera program and changes necessary to incorporate electric field effects and charged species interactions.

2.1.1 Base Physics

In order to properly model flame, a chemical kinetics and transport program must satisfy the same physical rules as a real-world flame. This goal requires that certain conservation equations must be adhered to when finding a solution to a simulated flame object. The conservation equations built into the original version of Cantera include Mass, Species, and Energy conservation as given below (Kee et al., 2003).

$$\frac{d}{dx}(\rho u) = 0 \tag{2-1}$$

$$\rho u \frac{dY_i}{dx} + \frac{d}{dx} J_i = \dot{\omega}_i W_i \tag{2-2}$$

$$\rho u c_p \frac{dT}{dx} + \frac{dq}{dx} + \left(\sum_i J_i c_{p,i} \right) \frac{dT}{dx} = -\sum_i \dot{\omega}_i h_i W_i \tag{2-3}$$

In equation 2-2 and 2-3, the terms $\dot{\omega}_i$ and J_i are given by equations 2-4 (Kee, 2003) and 2-5 (Eraslan et al., 1988) respectively.

$$\dot{\omega}_i = \sum_j \nu_{ij} \dot{\Omega}_j \tag{2-4}$$

$$J_i = \rho Y_i z_i \mu_i E_x - D_{m,i} \frac{W_i}{W_m} \frac{dX_i}{dx} \tag{2-5}$$

Equation 2-1 is used to maintain mass conservation. In Cantera, this is accomplished by maintaining a constant mass flux at each grid point in a one-dimensional flame. Each grid point represents a slice of the flame at a specific distance from the burner. Data expressing the state of the combustion mixture is stored at each grid point including species mass fractions and temperature.

Equation 2-2 represents the conservation of species within a combustion reaction. Molecules of a species may be created or eliminated through the use of balanced chemical reactions, but the total number of atoms of each element comprising the molecules must remain the same. As illustrated in this equation, the total mass of a given species may increase or decrease as a result of chemical reactions, given by the term on the right-hand-side of equation. The mass of species at any grid point is affected by the gas velocity, which is accounted for in first term on the left-hand-side of equation 2-2, and diffusion as shown in the second term of the same equation.

The final pre-packaged conservation equation is for the energy. Equation 2-3 accounts for movement of thermal energy within the flame due to bulk motion, thermal conduction and diffusion in the three terms on the left-hand-side, respectively. The term on the right-hand-side corresponds to heat generation through chemical reactions.

Equation 2-4 provides the production rate of each chemical species. This production rate is based on the results of all reactions which create or destroy molecules

of each simulated species. A more in-depth description of reaction rates can be found in section 2.2.1 or in the document *Defining Phases and Interfaces* (Goodwin, 2003).

Equation 2-5 is used to compute the mass flux of each species at each grid point within a simulation. Two major processes are responsible for mass flux within the reaction zone, diffusion flux and electrostatic drift flux. The first process, which is given in the standard version of Cantera, involves movement of molecules through flux of species due to relative differences in gas composition. The results of the effect are computed in the second term on the right-hand-side of equation 2-5. The second source of mass flux is drift of charged species caused by an applied electric field, which was not actually included in the base version of Cantera.

2.1.2 Modifications to Cantera

As discussed in section 2.1, significant modification to Cantera was required in order to calculate the effects of applied electric fields and charged species. The most significant modification was the addition of one governing equation (Pederson et al., 1993).

$$\varepsilon_0 \frac{dE_x}{dx} = \left(N_A e\right)\sum_i z_i \frac{\rho Y_i}{W_i} \tag{2-6}$$

$$E_x = -\frac{d\Phi}{dx} \tag{2-7}$$

Equation 2-6 represents Gauss's Law. Using Gauss's Law allows for the electric potential to be calculated at each grid point within a modeled flame object. This equation links the total charge at each grid point to the change in electric field at each grid point. The net charge at each grid point is found in the right hand side of equation 2-6 by finding the summation of the number of each charge carriers multiplied by each carrier's

respective charge. Also, the permittivity of a gas mix is approximated to be equivalent to that of free space. However, although the modified Cantera stores a values for E_x, values of Φ are used for calculations. The electric field, E_x, can be calculated from the electric potential, Φ, using Equation 2-7.

2.2 Reaction Mechanism Overview

A Reaction Mechanism is the set of information concerning possible elements, chemical species, and chemical reactions within a combustion fluid mixture. A wide variety of reactions and species are possible during combustion, but it is not feasible to model every reaction and species perfectly. As a result, reaction mechanisms are actually subsets of the real chemical processes within a reaction zone. It is possible to produce mechanisms for certain types of conditions such as rich or lean fuel mixes and use of specific fuels by adjusting reaction rates within the mechanism to fit the desired conditions, which the authors of GRIMech 3.0 have done to improve performance in calculating lean flame composition.

Three main types of information are required for a reaction mechanism. These types of information include thermodynamic properties of each species, molecular transport properties for each species, and a list of all reactions including reactants, products, and reaction rates. These properties will be discussed in more detail in sections 2.2.1 through 2.2.3.

Cantera uses two file formats for reaction mechanism files. The first format is the Cantera "cti" file format. "cti" files are actually valid Python scripts, but are also meant to be fairly simple for a human operator to interpret. The second type of format used by Cantera is the "ctml" format. The "ctml" file format is a variation of the XML format,

and it is meant to be easily parsed for use by computer. Cantera automatically generates "ctml" files from "cti" files, making direct manipulation of "ctml" files by a human operator unnecessary. As a result, the "cti" format will be used in all mechanism examples.

2.2.1 Reactions and Rates

Although Cantera can handle many types of reactions, only three types have been used in the reaction mechanisms assembled for this project. These types are homogeneous, three body, and falloff reactions. Because this topic is already well documented, only a short overview is provided here. More information on reactions and rates in Cantera is available in section 4.3 of the document, *Defining Phases and Interfaces* (Goodwin, 2003).

All three reaction types used include entries for the reaction equation. For the reaction equation, a string representing a balanced chemical equation is used. This equation string is delimited by spaces and uses the symbols "=" or "<=>" for reversible reactions or "=>" for nonreversible reactions (Goodwin, 2005: *Defining Phases and Interfaces*).

In addition to the balanced chemical equation, all reactions must include an entry for the reaction rate coefficient. Cantera assumes all rate coefficients to be in the form of a modified Arrhenius function as illustrated in equation 2-8.

$$k_f(T) = AT^n \exp\left(-E/RT\right) \tag{2-8}$$

The simplest of the three types of reactions used is the homogeneous pressure-independent reaction. For this type of entry, Cantera uses the "reaction" type in the "cti"

file, and only the equation and rate coefficients are required for a full entry as shown in Figure 2-1.

```
reaction(  "CH4 + O => CH3 + OH",  [4.07000E+14, 0, 7040])
```

Figure 2-1: Example of "cti" File Reaction Entry

The next most common reaction type in the studied mechanisms is the three body reaction. This type of reaction involves the use of a molecule of a species which does not react in addition to the reacting species. This non-reacting species provides or absorbs energy which stabilizes the reaction. In the chemical equation for this reaction type, the third body is listed simply as 'M', and the rate coefficients are still required. In addition to the required arguments, an optional string may be provided which gives the collision efficiencies for different third-body molecules. An example of the listing for this type of reaction is provided in Figure 2-2.

```
three_body_reaction("H2O2 + M => OH + OH + M", [1.692E+24, -2.00, 202.29],
        efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75 O2:0.4 N2:0.4 ")
```

Figure 2-2: Example of "cti" File Three Body Reaction Entry

The final type of reaction included is the Lindemann falloff reaction. This type of reaction differs from the previous two in that this reaction is pressure dependent. These reactions have two sets of rate coefficients, one for pressures approaching zero and one for pressures approaching infinity. Equations 2-9 and 2-10 show how these two sets of reaction rate coefficients are used to derive a rate coefficient for the reaction equation (Goodwin, 2005: *Defining Phases and Interfaces*). Figure 2-3 gives an example of an entry for this type of reaction within a Cantera "cti" file.

$$P_r = \frac{k_0[M]}{k_\infty} \tag{2-9}$$

$$k_f(T, P_r) = k_\infty \left(\frac{P_r}{1 + P_r} \right) \tag{2-10}$$

```
falloff_reaction("CH3 + H (+ M) => CH4 (+ M)",
                 kf = [2.108E+14, 0.00, 0.00],
                 kf0 = [6.257E+23, -1.80, 0.00])
```

Figure 2-3: Example of Cantera "cti" File Entry for a Lindemann Falloff Reaction

2.2.2 Thermodynamic Properties

The thermodynamic properties of the species within a reaction mixture used by Cantera include values for specific heat, entropy and enthalpy at range of temperatures. Cantera can use a few different formats for storing the thermodynamic data. These formats include the old NASA thermodynamic polynomial format and constant specific heat format.

The old NASA thermodynamic polynomial format has been used as the preferred format because it can be used to interpolate each of the thermodynamic properties at varying temperatures within a defined range rather than performing approximations with a single set of constants. The NASA polynomial format was produced and documented by Bonnie McBride and Sanford Gordon of NASA's Glen Research Center, and detailed information can be found for this format in NASA's documentation (Gordon et al, 1994).

Rather than holding exact values for each of the thermodynamic properties, the old NASA thermodynamic polynomial format stores two sets of seven coefficients for polynomial interpolation of the thermodynamic data. Each set of seven coefficients corresponds to a interpolation in one temperature range. The valid temperature range for each polynomial must also be provided in the entry. An example of a NASA thermodynamic data entry can be seen in Figure 2-4.

```
thermo = (
       NASA( [  200.00,  1000.00], [  5.149876130E+00,  -1.367097880E-02,
                4.918005990E-05,  -4.847430260E-08,   1.666939560E-11,
               -1.024664760E+04,  -4.641303760E+00] ),
       NASA( [ 1000.00,  3500.00], [  7.485149500E-02,   1.339094670E-02,
               -5.732858090E-06,   1.222925350E-09,  -1.018152300E-13,
               -9.468344590E+03,   1.843731800E+01] )
             )
```

Figure 2-4: NASA polynomial entry for Methane.

In Figure 2-4, the thermo encloses two NASA polynomials, one for a high temperature range and the other for a low range. Each NASA entry contains one set of NASA polynomial constants. The temperature range for each polynomial is provided in the first grouping within square brackets in each NASA entry. The second grouping contains the seven polynomial coefficients for the entry. For example, in Figure 2-4, the temperature range for the first NASA polynomial is 200.0 to 1000.0 Kelvin, and the seven polynomial coefficients for the first temperature range are the seven numbers enclosed in square brackets following the temperature range.

The non-dimensional values for specific heat, entropy, and enthalpy for any temperature within the valid range of the polynomial can be found using the following equations (Gordon et al, 1994).

$$\frac{c_{p0}}{R} = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{2-11}$$

$$\frac{H_0}{RT} = a_1 + \frac{a_2}{2T} + \frac{a_3}{3T^2} + \frac{a_4}{4T^3} + \frac{a_5}{5T^4} + \frac{a_6}{T} \tag{2-12}$$

$$\frac{S_0}{R} = a_1 \ln(T) + a_2 T + \frac{a_3}{2} T^2 + \frac{a_4}{3} T^3 + \frac{a_5}{4} T^4 + a_7 \tag{2-13}$$

In order to convert the non-dimensional values resulting from equations 2-11 and 2-13 to the proper units simply multiply by the form of the ideal gas constant with the

desired units. For equation 2-12, multiply by the ideal gas constant and the temperature in Kelvin to convert to the proper units.

A Python program to plot curves for the thermodynamic properties given the NASA polynomials in Chemkin format is provided in Appendix C. Figures 2-5 through 2-7 provide an example of the polynomial curves for methane as output by the previously mentioned program. This program was written with the purpose of checking NASA polynomials for possible discontinuities between temperature ranges, which could result in erroneous results and failure of the model to converge on a solution.

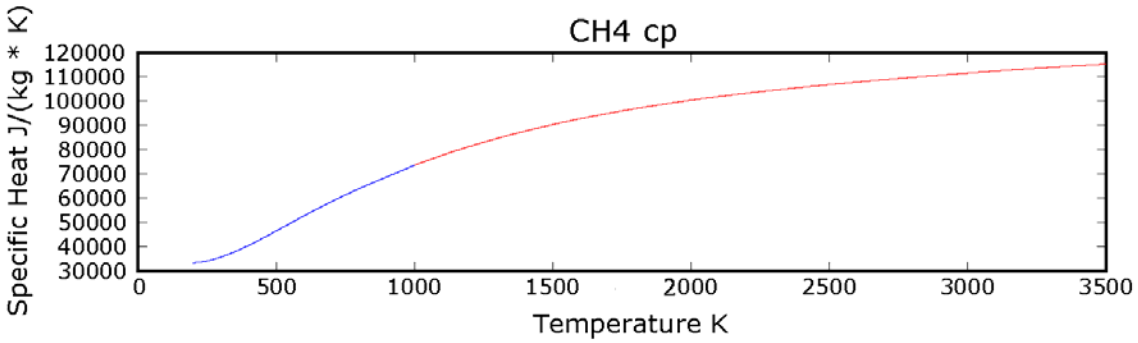

Figure 2-5: Plot of the Specific Heat of Methane Calculated from the NASA Polynomial

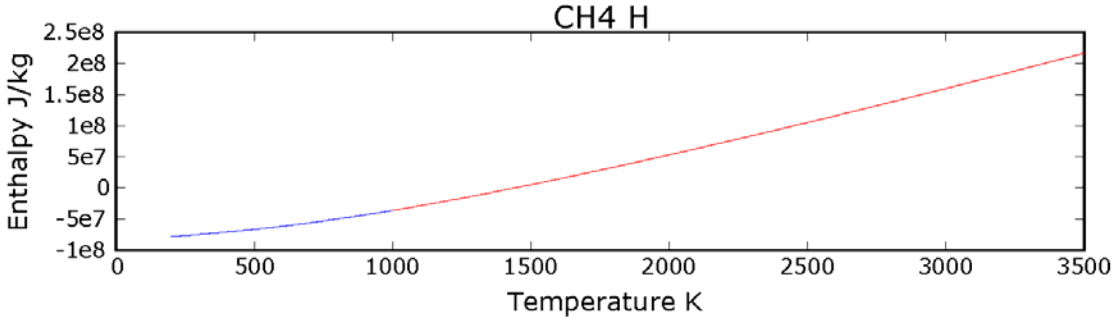

Figure 2-6: Plot of the Enthalpy of Methane Calculated from the NASA Polynomial

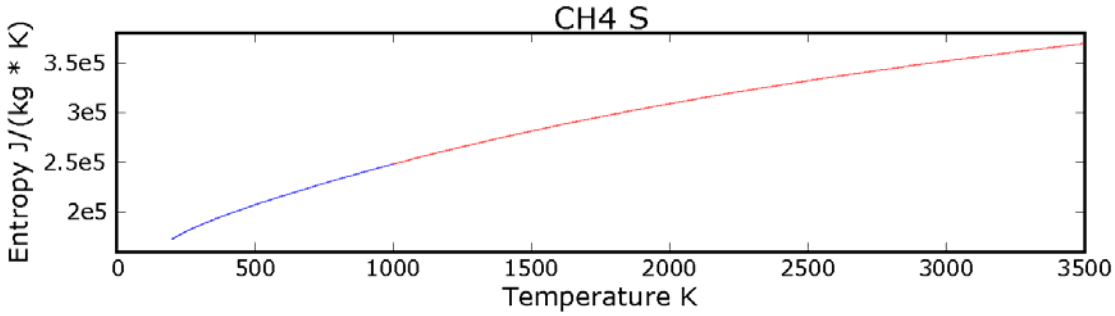

Figure 2-7: Plot of the Entropy of Methane Calculated from the NASA Polynomial

A second format used by Cantera for the thermodynamic properties is the constant specific heat format. This format allows for the use of a constant value for the specific heat of a given species, and entropy and enthalpy consistant with the constant specific heat and base values of entropy and enthalpy given at a single reference temperature using the following equations.

$$\frac{H_0(T)}{RT} = H_0(T_0) + c_{p0}(T - T_0) \tag{2-14}$$

$$\frac{S_0(T)}{R} = S_0(T_0) + c_{p0} \ln\left(T/T_0\right) \tag{2-15}$$

Because the use of constants is not as accurate as the NASA polynomial format, it was only used when a NASA polynomial could not be found for a desired chemical species. Figure 2-8 shows an example of a "cti" file entry for constant thermodynamic data.

```
thermo = const_cp(t0 = 298.15,
                  h0 = (255.0, 'cal/mol'),
                  s0 = (59.0, 'cal/K/mol'),
                  cp0 = (14.1, 'cal/K/mol')),
```

Figure 2-8: Constant thermodynamic data entry for C_3H_3+

2.2.3 Molecular Transport Properties

The final type of species data which Cantera uses is the molecular transport data. The purpose of the transport data is to calculate coefficients such as viscosity and species

diffusivities which help model the effects of non-reactive species interactions. In order to calculate these properties, Cantera uses a set of at least four properties for each species. These properties include the species geometry, Lennard-Jones collision diameter, Lennard-Jones well depth, and the rotational relaxation collision number at 298 Kelvin. A more detailed entry may be used as described in figure 4.6 of the document, *Defining Phases and Interfaces* (Goodwin, 2003).

2.3 Tested Published Reaction Mechanisms

As part of this research, a sampling of published reaction mechanisms was used. These mechanisms include the Hiensohn, Jones and Becker mechanism for methane, GRI 3.0 mechanism, a mechanism by Norbert Peters, a mechanism published by Timothy Pederson and Robert C. Brown, and modified forms of the GRI 3.0 and Norbert Peters mechanisms.

2.3.1 Hiensohn, Jones, and Becker

The Hiensohn, Jones and Becker mechanism is a mechanism developed for the study of opposed-jet diffusion flames. The original purposes of this mechanism were to develop a model of the opposed-jet diffusion flame which possesses a good treatment of fluid flow, but also contains a realistic set of chemical reactions and transport properties, and to investigate the effect of a simplified model of an electric field upon such a flame (Jones et al., 1972).

The reason for using this mechanism for the CCADS project is that the mechanism is fairly small, and thus requires less computational time than the more extensive mechanisms making it ideal for model developement. This mechanism contains nineteen species and thirty-one reactions, which is significantly less than the

other three mechanisms used in this research. Also, this mechanism contains a simple model for charged particle chemistry which may be added with caution to larger neutral species reaction mechanisms. The full mechanism file is provided in the Appendix B, but the included charge particle reactions are given in the chemical equations below (Jones et al., 1972).

$$CH + O \rightarrow HCO^{+} + e^{-} \quad (2\text{-}16)$$

$$HCO^{+} + H_2O \rightarrow CO + H_3O^{+} \quad (2\text{-}17)$$

$$H_3O^{+} + e^{-} \rightarrow H_2O + H \quad (2\text{-}18)$$

In the above equations, the presence of three charged species is given. Equation 2-16 shows one possible path for the formation of electrons. Equation 2-17 shows the formation of the HCO^{+} ion, which is used in the production of the more abundant H_3O^{+} ion in equation 2-18.

2.3.2 GRI and Modified GRI

The GRI 3.0 mechanism is an extensive and wide ranging mechanism developed by the Gas Research Institute. Cantera includes a hard-coded version of the GRI mechanism within its source code, but a full version of this mechanism in Chemkin II format is freely available for download on the GRI Mechanism authors' webpage. The unmodified version of GRI 3.0 contains fifty-three chemical species and 325 reactions, which causes it to be computationally expensive. However, GRI 3.0 contains no charged species, and as a result, cannot be used to model charged particle generation and movement without modification. The only modification made to allow charged species chemistry was the addition of the three charged species transport and thermodynamic data plus three charged species reactions provided for the Jones, Heinsohn and Becker mechanism described in section 2.2.4 of this document.

A full listing of the Cantera "cti" file for the modified version of the GRI 3.0 mechanism is given is Appendix B.

2.3.3 Peters and Modified Peters

The Peters mechanism is a mechanism designed to represent neutral species reactions within premixed methane-air flames. The unmodified version of this reaction mechanism contains thirty-one species and 107 reactions, which is more complex than the Jones, Becker and Heinsohn mechanism, but not as detailed as the GRI 3.0 mechanism. The modified version of this mechanism contains the charges species and reactions from the Jones, Becker and Heinsohn mechanism discussed in section 2.2.4 (Peters, 1992).

A full listing of the Cantera "cti" file for the modified version of the Peters mechanism is given is Appendix B.

2.3.4 Pedersen and Brown

The Pedersen and Brown mechanism was designed for the study of electric field effects within methane flames in order to attempt to control blowoff limits, flame speed and soot formation by the application of electric fields (Pedersen et al, 1993). The Pedersen and Brown mechanism contains eighty-five reactions and total of thirty-five chemical species including five charged species. This mechanism contains many additional charged species reactions which were not included in the three previously mentioned reaction mechanisms. These charged species reactions are given below.

$$CH + O \rightarrow CHO^+ + e^- \quad (2\text{-}19)$$

$$CH^* + O \rightarrow CHO^+ + e^- \quad (2\text{-}20)$$

$$CHO^+ + H_2O \rightarrow H_3O^+ + CO \quad (2\text{-}21)$$

$$H_3O^+ + C_2H_2 \rightarrow C_2H_3O^+ + H_2 \quad (2\text{-}22)$$

$$CHO^+ + CH_2 \rightarrow CH_3^+ + CO \quad (2\text{-}23)$$

$$H_3O^+ + CH_2 \rightarrow CH_3^+ + H_2O \quad (2\text{-}24)$$

$$CH_3^+ + C_2H_2 \rightarrow C_3H_3^+ + H_2 \quad (2\text{-}25)$$

$$C_3H_3^+ + H_2O \rightarrow C_2H_3O^+ + CH_2 \quad (2\text{-}26)$$

$$CH_3^+ + CO_2 \rightarrow C_2H_3O^+ + O \quad (2\text{-}27)$$

$$H_3O^+ + e^- \rightarrow H_2O + H \quad (2\text{-}28)$$

$$C_3H_3^+ + e^- \rightarrow C_2H_2 + CH \quad (2\text{-}29)$$

$$CH_3^+ + e^- \rightarrow CH_2 + H \quad (2\text{-}30)$$

In the above equations, CH* represents electronically excited CH molecules. The electronically excited CH molecules react 2000 times faster than non-excited CH molecules (Pederson and Brown, 1993). Also, equation 2-29 has been modified from the form provided by Pederson and Brown because the original document listed some of the reaction products simply as "products" rather than providing a full balanced chemical equation.

For a complete listing of the Cantera "cti" format mechanism file, see Appendix B.

2.4 Current Calculations

During a simulation run, an output file to store simulation current values may be created simply by specifying a filename for the results. The file format for these types of files is given in detail in Appendix D-3. The column labeled "current" containied within the output file is actually a current density (A/m^2), but the real current (A) can be derived simply by multiplying the current density by the burner's area because this simulation represents a one-dimensional line through an axialsymetric system, resulting in uniform current density.

Chapter Three: Verification of Model

3.1 Existing Experimental Data

In order to verify this modeling software, simulations were run for which experimental data already existed. The source of the experimental data for comparison was Christopher Hill's research at NETL in the first half of 2004. His research included experimental measurements of currents within both methane-air and synthesis gas-air premixed flat flames.

These experiments were conducted with the purpose of aiding the understanding of flame ionization and transport. Both methane and methane-doped synthesis gas were tested using similar experimental setups resulting in V-I curves found by taking measurements with varying applied electrical potentials across a flame. In addition, a flat flame burner was used for the experimentation, which provided for the use of a computationally simpler one-dimension model because the properties of a flat flames are the same above nearly the entire burner area.

3.1.1 NETL Spring 2004 Flat Flame Burner Experiments

Christopher Hill, a former intern at the Morgantown NETL location, conducted experiments on lean combustion with both methane and methane doped synthesis gas fuels. These experiments were conducted on a porous disk flat flame burner mounted on an adjustable stage control, which positioned the burner in relation to a stationary grid electrode mounted above the burner. This setup is illustrated in Figures 3-1 through 3-3. As shown if Figure 3-3, an electric field was applied through the flame by using the burner itself as a second electrode and applying an electric potential between the grid electrode and the burner.

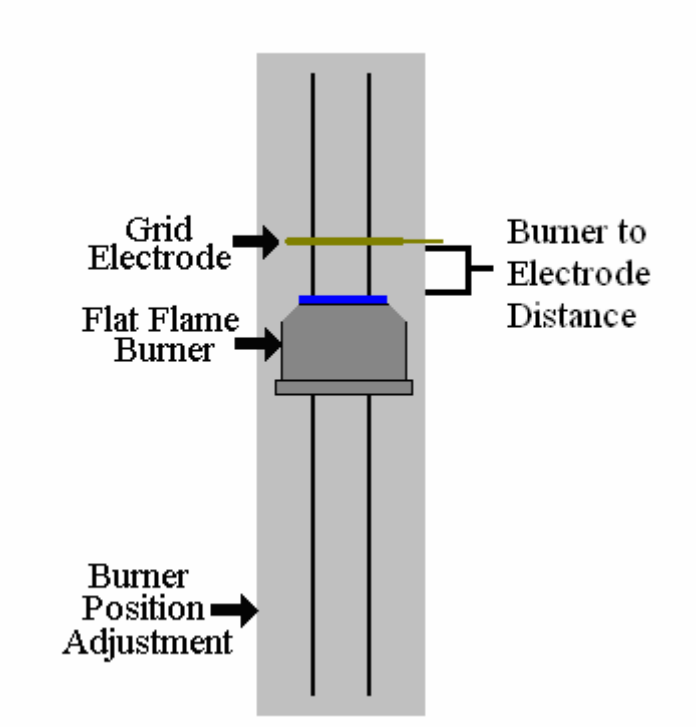

Figure 3-1: Illustration of Flat Flame Burner Mounted on an Adjustable Stage Control

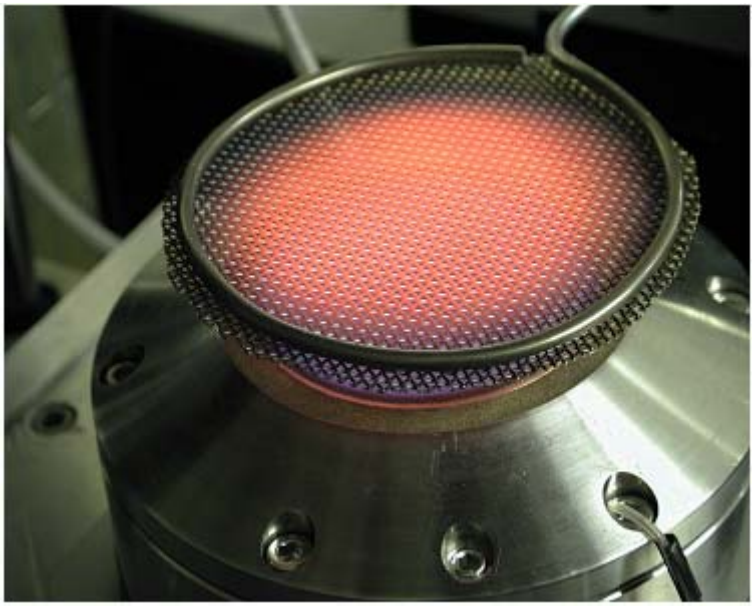

Figure 3-2: Flat Flame Burner with Grid Electrode

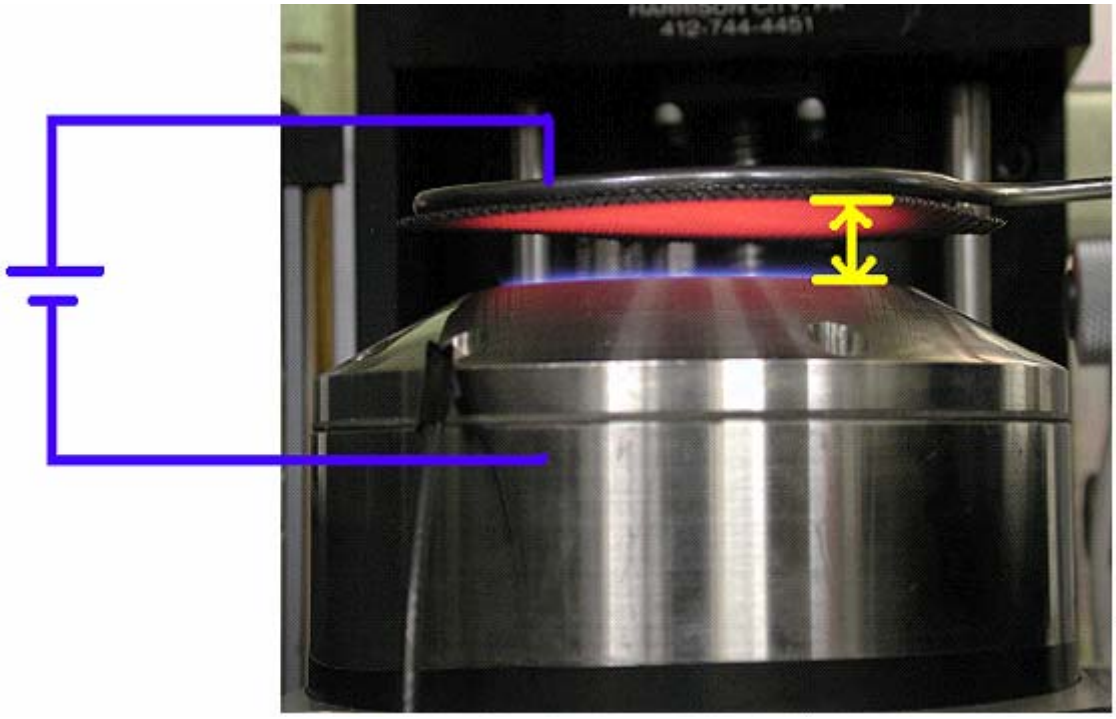

Figure 3-3: Simplified Electrical Connection for Flat Flame Burner and Grid Electrode

Table 3-1 states the mixture conditions for Chris Hill's methane experiments.

Experiment Number	Air Flow (SLPM)	Air Flow (kg/s)	Methane (SLPM)	Methane (kg/s)	Burner-Electrode Distance (cm)
1-1	10.00	0.000215483	1.00	1.19×10^{-5}	1.60
1-2	11.00	0.000237032	1.00	1.19×10^{-5}	1.60
1-3	12.00	0.00025858	1.00	1.19×10^{-5}	1.60
1-4	13.00	0.000280128	1.00	1.19×10^{-5}	1.60
1-5	14.00	0.000301677	1.00	1.19×10^{-5}	1.60
1-6	15.00	0.000323225	1.00	1.19×10^{-5}	1.60
2-1	10.00	0.000215483	1.00	1.19×10^{-5}	0.20
2-2	11.00	0.000237032	1.00	1.19×10^{-5}	0.20
2-3	12.00	0.00025858	1.00	1.19×10^{-5}	0.20
2-4	13.00	0.000280128	1.00	1.19×10^{-5}	0.20
2-4	14.00	0.000301677	1.00	1.19×10^{-5}	0.20
2-6	15.00	0.000323225	1.00	1.19×10^{-5}	0.20
3-1	10.00	0.000215483	1.00	1.19×10^{-5}	1.60
3-2	10.00	0.000215483	0.91	1.09×10^{-5}	1.60
3-3	10.00	0.000215483	0.83	9.91×10^{-6}	1.60
3-4	10.00	0.000215483	0.77	9.19×10^{-6}	1.60
3-5	10.00	0.000215483	0.71	8.47×10^{-6}	1.60
3-6	10.00	0.000215483	0.67	8.00×10^{-6}	1.60

Table 3-1: Methane Experimental Conditions

Tables 3-2 and 3-3 state the mixtures used for the synthesis gas experiments.

Experiment	Air Flow	CO Flow	H2 Flow	CH4 Flow	N2 Flow	Burner-Electrode Distance (cm)
1	10.00	1.2361	9.99E-01	4.00E-02	3.95E-01	2.00
2	10.00	1.2361	9.99E-01	8.00E-02	3.95E-01	2.00
3	10.00	1.2361	9.99E-01	1.20E-01	3.95E-01	2.00
4	10.00	1.2361	9.99E-01	1.40E-01	3.95E-01	2.00
5	10.00	1.2361	9.99E-01	1.60E-01	3.95E-01	2.00

Table 3-2: Synthesis Gas Experimental Conditions in SLPM

Experiment	Air Flow	CO Flow	H2 Flow	CH4 Flow	N2 Flow	Burner-Electrode Distance (cm)
1	1.97E-04	2.36E-05	1.37E-06	4.37E-07	7.53E-06	2.00
2	1.97E-04	2.36E-05	1.37E-06	8.74E-07	7.53E-06	2.00
3	1.97E-04	2.36E-05	1.37E-06	1.31E-06	7.53E-06	2.00
4	1.97E-04	2.36E-05	1.37E-06	1.53E-06	7.53E-06	2.00
5	1.97E-04	2.36E-05	1.37E-06	1.75E-06	7.53E-06	2.00

Table 3-3: Synthesis Gas Experimental Conditions in kg/s

Figures 3-4 through 3-7 are plots that show the results of Christopher Hill's experiments. These results will be compared with the simulation results in Chapter 4 of this document. Tables 3-4 and 3-5 show slopes and values of the experimental current curve up to ninety percent of the maximum current.

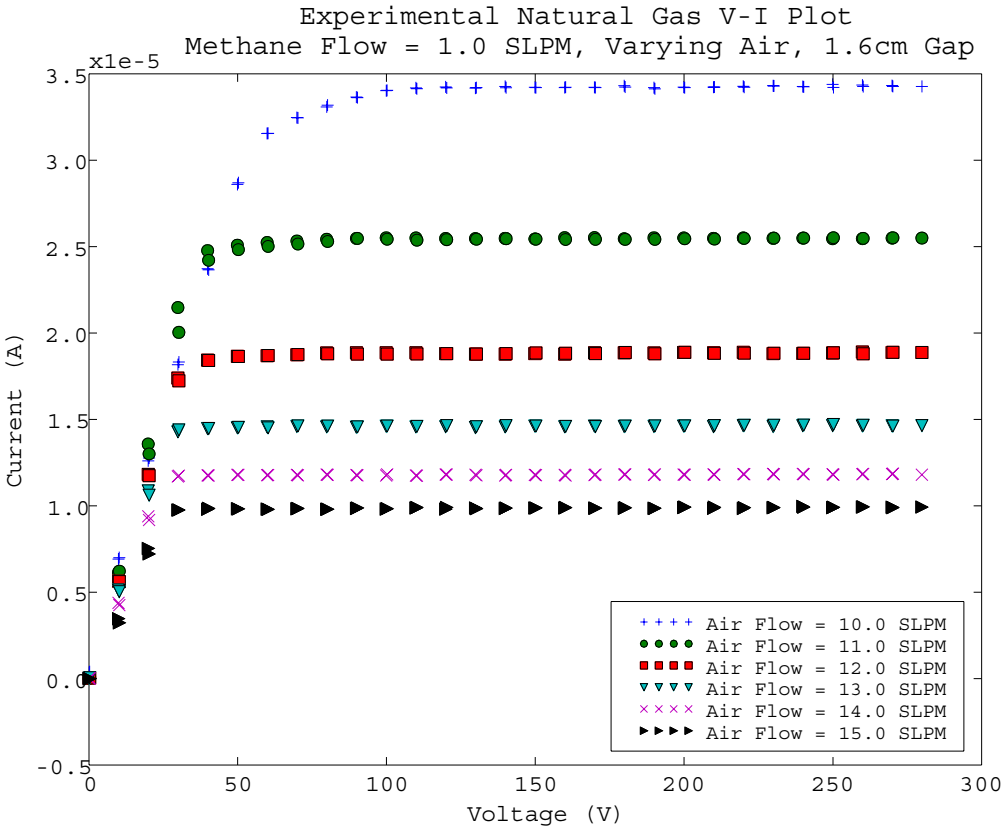

Figure 3-4: Experimental Methane Data with Constant Air Flow and Forward Voltage

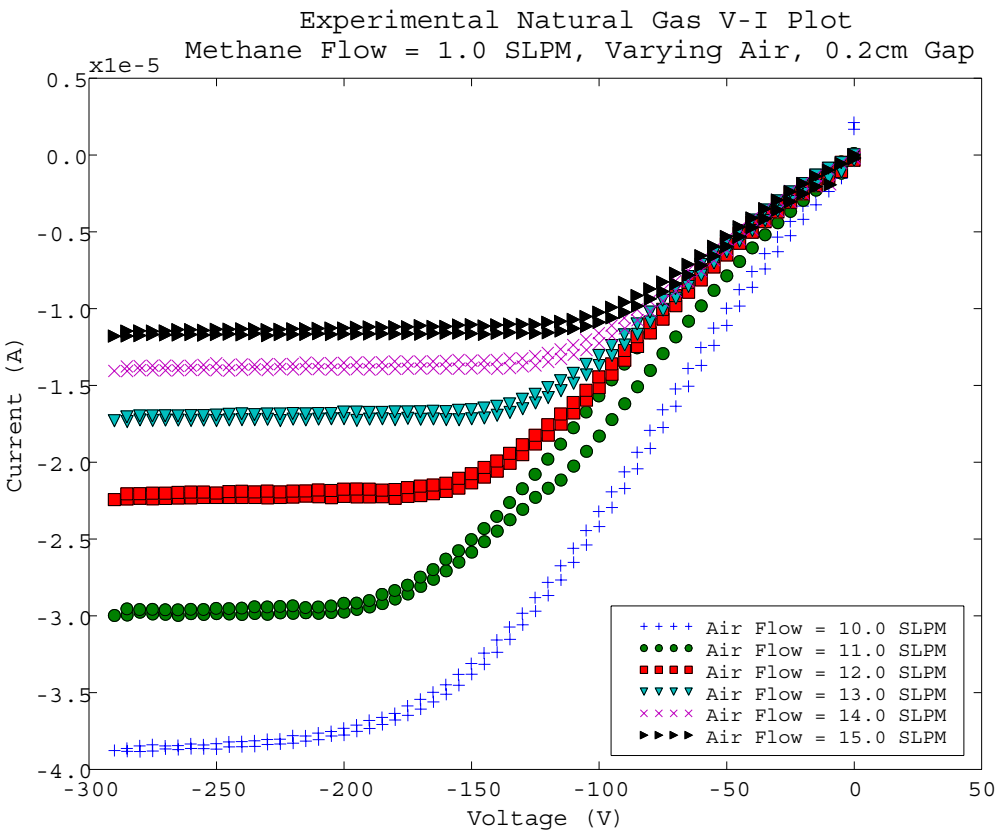

Figure 3-5: Experimental Methane Data with Constant Air Flow and Negative Voltage

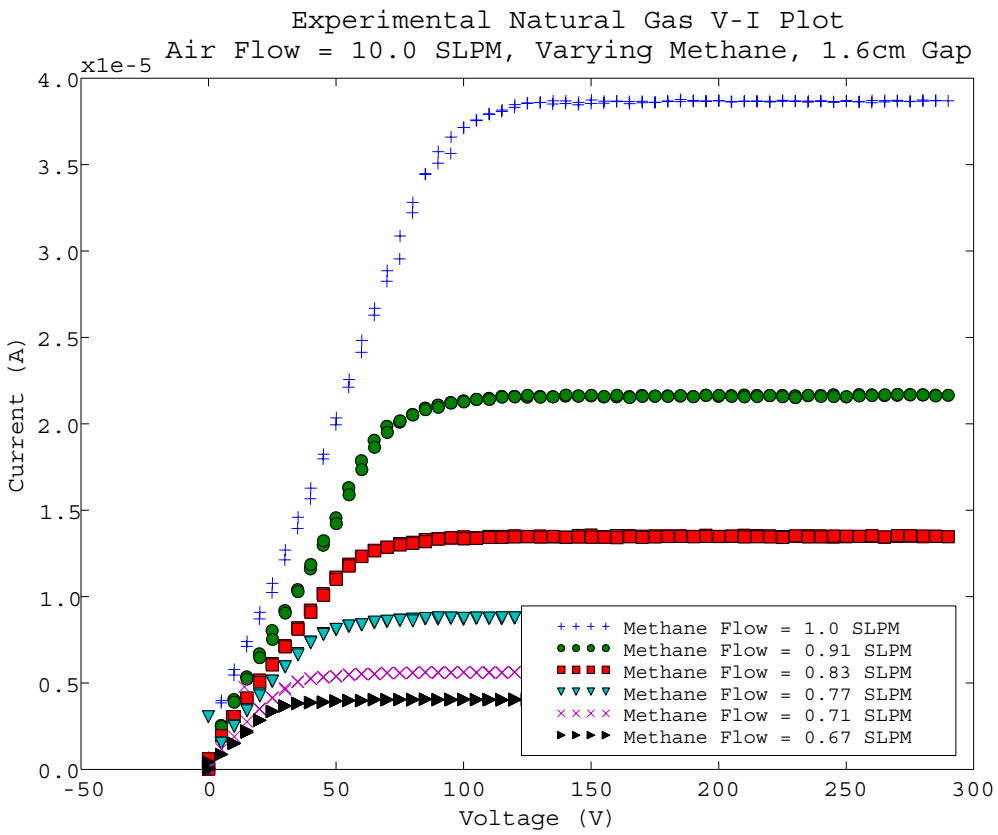

Figure 3-6: Experimental Methane Data with Constant Methane Flow and Forward Voltage

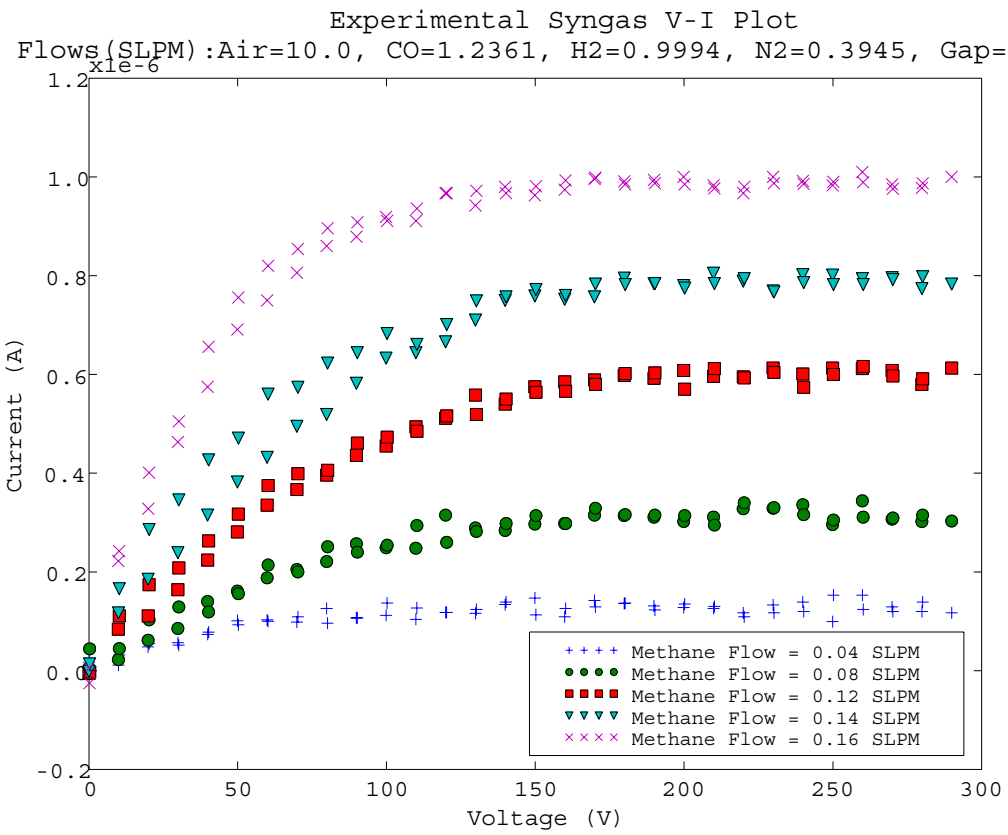

Figure 3-7: Experimental Synthesis Gas Data with Varying Methane Concentrations Added

Experiment Number	Experimental Methane Combustion V-I Curve Characteristics to 90% of Peak Current		
	Slope	Current	Voltage
1-1	5.2152e-007	3.1549e-005	59.876
1-2	6.2169e-007	2.4767e-005	39.878
1-3	5.8133e-007	1.7386e-005	29.894
1-4	4.7797e-007	1.4281e-005	29.894
1-5	3.9118e-007	1.1692e-005	29.894
1-6	3.2688e-007	9.7458e-006	29.894
2-1	2.301e-007	-3.511e-005	-159.96
2-2	1.6769e-007	-2.7077e-005	-159.95
2-3	1.4466e-007	-2.058e-005	-139.95
2-4	1.287e-007	-1.5781e-005	-119.95
2-5	1.1904e-007	-1.2695e-005	-104.95
2-6	1.071e-007	-1.0889e-005	-99.949
3-1	3.8242e-007	3.5078e-005	89.953
3-2	2.8371e-007	1.9859e-005	69.943
3-3	1.9529e-007	1.2334e-005	59.947
3-4	1.0109e-007	8.102e-006	49.945
3-5	1.3053e-007	5.2605e-006	39.949
3-6	1.0804e-007	3.7985e-006	34.946

Table 3-4: Experimental Methane Combustion V-I Curve Characteristics to 90% of the Maximum Current

Run Number	Experimental Synthesis Gas Combustion V-I Curve Characteristics to 90% of Peak Current		
	Slope	Current	Voltage
1	9.99E-10	1.39E-07	140.18
2	2.58E-09	3.15E-07	119.85
3	4.34E-09	5.58E-07	129.85
4	5.77E-09	7.49E-07	130.19
5	9.15E-09	9.19E-07	99.866

Table 3-5: Experimental Synthesis Gas Combustion V-I Curve Characteristics to 90% of the Maximum Current

3.2 Mechanism Refinement Study

The mechanism refinement study performed as part of this research has been conducted by performing simulations using a variety of published reaction mechanisms, and comparing simulation results with the experimental data obtained at NETL. Each of the mechanisms used for this research is discussed in Chapter 2, and a full listing of the Cantera "cti" format mechanism files appear in Appendix B.

Each mechanism was chosen based on certain characteristics such as mechanism size and amount of previous use. The Jones, Becker and Heinsohn mechanism was used because it provides built-in support for charged species interactions and its small size, which allows for expedited computation. The Pederson and Brown mechanism is an intermediate-sized reaction mechanism with a more extensive set of charged species than the Jones, Becker and Heinsohn mechanism. The original form of the Peters mechanism included only charge neutral species, but included comprehensive documentation of the purpose of each set of reactions for combustion simulation. GRI Mechanism 3.0 has been included because it is a commonly-used and well-refined mechanism for combustion simulation, but its large size generally results in significantly longer simulation times than with the smaller Jones, Becker, and Heinsohn mechanism.

By running each mechanism and comparing the results with the experimental results given in Section 3.1, it is possible to determine how well each mechanism performs in calculating electrical current within a flat flame. In addition, some potential problems with each mechanism can be observed using species mass or mole fractions which have not been recorded for the experimental flames, but are calculated in the simulated flame objects.

Chapter Four: Comparison of Experimental and Simulation Results

4.1 Methane Models

This section will give a comparison of the methane simulations with the experimental data explained in Chapter Three of this document. For the methane combustion simulation, four different reaction mechanisms were used:

1) GRI with Jones, Becker and Heinsohn charged species chemistry
2) Jones, Becker and Heinsohn
3) Pederson and Brown
4) Norbert Peters with Jones, Becker and Heinsohn charged species chemistry

All four of these reaction mechanisms are discussed in Chapter Two, and the full listings of the Cantera 'cti' files for each mechanism are given in Appendix B.

In addition to the use of multiple reaction mechanisms, three different constant electron mobilities were used with each reaction mechanism. These electron mobilities were 0.36, 0.4, and 0.44 m^2/Vs. These mobilities were selected by using the value of 0.4 m^2/Vs as the default electron mobility within the gas mixture as stated by Goodings et al. in "Current-Voltage Characteristics in a Flame Plasma", and taking testing at ±10% of this base value.

For this research, two properties of the V-I profiles were of prime importance. These properties include the slope of the curve from zero volts and continuing until the peak current level is approached. This slope equals the conductivity of the simulated flame. The second important feature is the peak current level, which is known as the saturation current. Both of these properties are illustrated in the following figure.

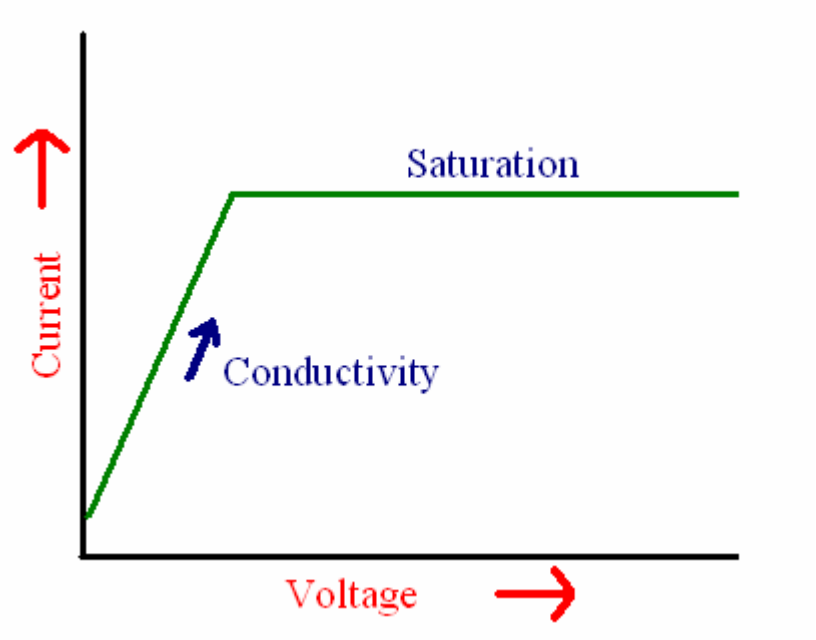

Figure 4-1: Simple Diagram of Saturation Current and Conductivity

The above figure also corresponds to the type of V-I curve generated in semiconductors using, in which carrier velocity increases until a maximum or saturation velocity is reached. This property for semiconductors is illustrated in Equation 1-3.

For the methane combustion experiments, experiment one represents increasing air with a forward bias. Experiment two is increasing air with a reverse bias, and experiment three is decreasing methane with a forward bias. In all three experiments, either air or methane is held constant.

4.1.1 GRI Mech 3.0

The modified GRIMech 3.0 was one of the two best mechanisms used in this research for predicting electrical characteristics of a burning natural gas flame. In addition, it is a highly researched, widely used mechanism for varying types of fossil fuel combustion, and as a result, tends to predict bulk properties of the flames such as flame position and temperature fairly well.

A brief check of the V-I curve plots in Appendix E show that overall, the prediction of the saturation currents, although not perfect, was fairly close in many cases. One notable feature in the V-I curves is that for the higher equivalence ratio tests, the saturation current is in very good agreement with the experimental results, but as the equivalence ratio is lowered, this mechanism has a tendancy to underpredict the saturation

However, the conductivity, illustrated by the V-I curve slopes, overpredicts the experimental measurements with all three of the tested mobility values. The error in conductivity decreases with the electron mobility, which indicates that a lower value for the mobility may lead to significantly better results. The reasonable predictions of saturation current coupled with the high predictions for conductivity result in the saturation current being reached at much lower voltages than shown in the experimental results.

The following tables show key properties of the V-I curve up to 90% of the saturation current for each of three tested mobility values.

Experiment Number	GRIMech 3.0 Methane Combustion Curve Characteristics to 90% of Peak Current with Mobility = 0.36 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	5.8420E-07	6.2680E-08	3.4372E-05	2.8230E-06	56	-3.876
1-2	6.3416E-07	1.2470E-08	2.0473E-05	-4.2940E-06	30	-9.878
1-3	7.4579E-07	1.6446E-07	1.3252E-05	-4.1340E-06	16	-13.894
1-4	7.3492E-07	2.5695E-07	9.2683E-06	-5.0127E-06	11	-18.894
1-5	6.6715E-07	2.7597E-07	7.1708E-06	-4.5212E-06	9	-20.894
1-6	5.6255E-07	2.3567E-07	5.5646E-06	-4.1812E-06	8	-21.894
2-1	1.9075E-06	1.6774E-06	-3.6781E-05	-1.6710E-06	-23	136.96
2-2	1.4851E-06	1.3174E-06	-2.1396E-05	5.6810E-06	-18	141.95
2-3	1.2513E-06	1.1066E-06	-1.3578E-05	7.0020E-06	-14	125.95
2-4	1.1210E-06	9.9230E-07	-9.3950E-06	6.3860E-06	-11	108.95
2-5	1.0306E-06	9.1156E-07	-7.0619E-06	5.6331E-06	-9	95.95
2-6	9.2959E-07	8.2249E-07	-5.7668E-06	5.1222E-06	-8	91.949
3-1	5.8420E-07	2.0178E-07	3.4372E-05	-7.0600E-07	56	-33.953
3-2	6.0448E-07	3.2077E-07	1.6395E-05	-3.4640E-06	25	-44.943
3-3	5.9640E-07	4.0111E-07	8.1473E-06	-4.1867E-06	12	-47.947
3-4	4.8862E-07	3.8753E-07	4.6745E-06	-3.4275E-06	8	-41.945
3-5	3.1236E-07	1.8183E-07	2.7262E-06	-2.5343E-06	7	-32.949
3-6	2.3066E-07	1.2262E-07	1.8354E-06	-1.9631E-06	6	-28.946

Table 4-1: V-I Curve Characteristics to 90% of Maximum Current Using GRIMech 3.0 with Electron Mobility of 0.36 m^2/Vs

Experiment Number	GRIMech 3.0 Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.40 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	6.3966E-07	1.1814E-07	3.4428E-05	2.8790E-06	51	-8.876
1-2	7.2509E-07	1.0340E-07	2.0427E-05	-4.3400E-06	26	-13.878
1-3	8.4554E-07	2.6421E-07	1.3249E-05	-4.1370E-06	14	-15.894
1-4	8.0666E-07	3.2869E-07	9.3561E-06	-4.9249E-06	10	-19.894
1-5	7.1039E-07	3.1921E-07	6.9446E-06	-4.7474E-06	8	-21.894
1-6	5.5937E-07	2.3249E-07	5.6134E-06	-4.1324E-06	8	-21.894
2-1	1.9104E-06	1.6803E-06	-3.6770E-05	-1.6600E-06	-23	136.96
2-2	1.4878E-06	1.3201E-06	-2.1388E-05	5.6890E-06	-18	141.95
2-3	1.2532E-06	1.1085E-06	-1.3571E-05	7.0090E-06	-14	125.95
2-4	1.1224E-06	9.9370E-07	-9.3891E-06	6.3919E-06	-11	108.95
2-5	1.0327E-06	9.1366E-07	-7.0585E-06	5.6365E-06	-9	95.95
2-6	9.3177E-07	8.2467E-07	-5.7643E-06	5.1247E-06	-8	91.949
3-1	6.3966E-07	2.5724E-07	3.4428E-05	-6.5000E-07	51	-38.953
3-2	6.8154E-07	3.9783E-07	1.6388E-05	-3.4710E-06	22	-47.943
3-3	6.4571E-07	4.5042E-07	8.1736E-06	-4.1604E-06	11	-48.947
3-4	4.8809E-07	3.8700E-07	4.7265E-06	-3.3755E-06	8	-41.945
3-5	3.3892E-07	2.0839E-07	2.6077E-06	-2.6528E-06	6	-33.949
3-6	2.2735E-07	1.1931E-07	1.8452E-06	-1.9533E-06	6	-28.946

Table 4-2: V-I Curve Characteristics to 90% of Maximum Current Using GRIMech 3.0 with Electron Mobility of 0.40 m^2/Vs

Experiment Number	GRIMech 3.0 Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.44 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	7.0403E-07	1.8251E-07	3.4334E-05	2.7850E-06	46	-13.876
1-2	8.1438E-07	1.9269E-07	2.0399E-05	-4.3680E-06	23	-16.878
1-3	9.1012E-07	3.2879E-07	1.3348E-05	-4.0380E-06	13	-16.894
1-4	8.7213E-07	3.9416E-07	9.2309E-06	-5.0501E-06	9	-20.894
1-5	7.0818E-07	3.1700E-07	7.0059E-06	-4.6861E-06	8	-21.894
1-6	5.5641E-07	2.2953E-07	5.6528E-06	-4.0930E-06	8	-21.894
2-1	1.9108E-06	1.6807E-06	-3.6744E-05	-1.6340E-06	-23	136.96
2-2	1.4902E-06	1.3225E-06	-2.1383E-05	5.6940E-06	-18	141.95
2-3	1.2555E-06	1.1108E-06	-1.3567E-05	7.0130E-06	-14	125.95
2-4	1.1245E-06	9.9580E-07	-9.3855E-06	6.3955E-06	-11	108.95
2-5	1.0345E-06	9.1546E-07	-7.0556E-06	5.6394E-06	-9	95.95
2-6	9.3337E-07	8.2627E-07	-5.7621E-06	5.1269E-06	-8	91.949
3-1	7.0403E-07	3.2161E-07	3.4334E-05	-7.4400E-07	46	-43.953
3-2	7.5136E-07	4.6765E-07	1.6528E-05	-3.3310E-06	20	-49.943
3-3	6.9711E-07	5.0182E-07	8.1183E-06	-4.2157E-06	10	-49.947
3-4	4.8657E-07	3.8548E-07	4.7664E-06	-3.3356E-06	8	-41.945
3-5	3.3653E-07	2.0600E-07	2.6251E-06	-2.6354E-06	6	-33.949
3-6	2.2319E-07	1.1515E-07	1.8537E-06	-1.9448E-06	6	-28.946

Table 4-3: V-I Curve Characteristics to 90% of Maximum Current Using GRIMech 3.0 with Electron Mobility of 0.44 m^2/Vs

With this mechanism, the saturation current using all three mobilities appears to be nearly the identical. Although the simulated saturation current are fairly close to the experimental values in many cases, there is a tendency for the underprediction of saturation current as the equivalence ratio is lowered.

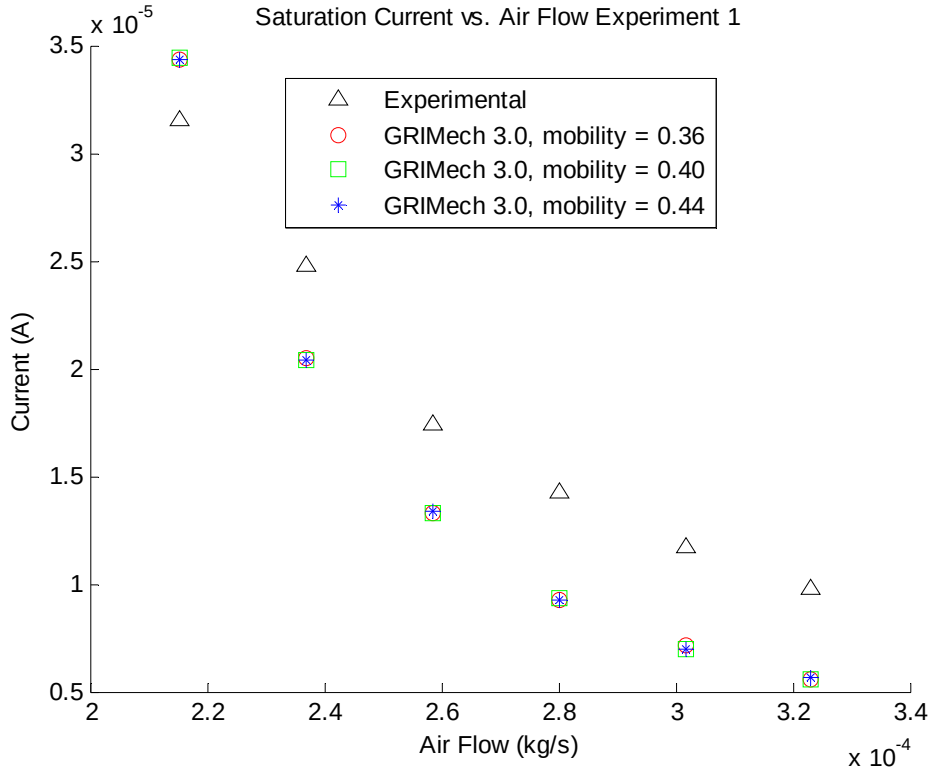

Figure 4-2: Saturation Currents Using GRIMech3.0 with Methane in Experiment 1

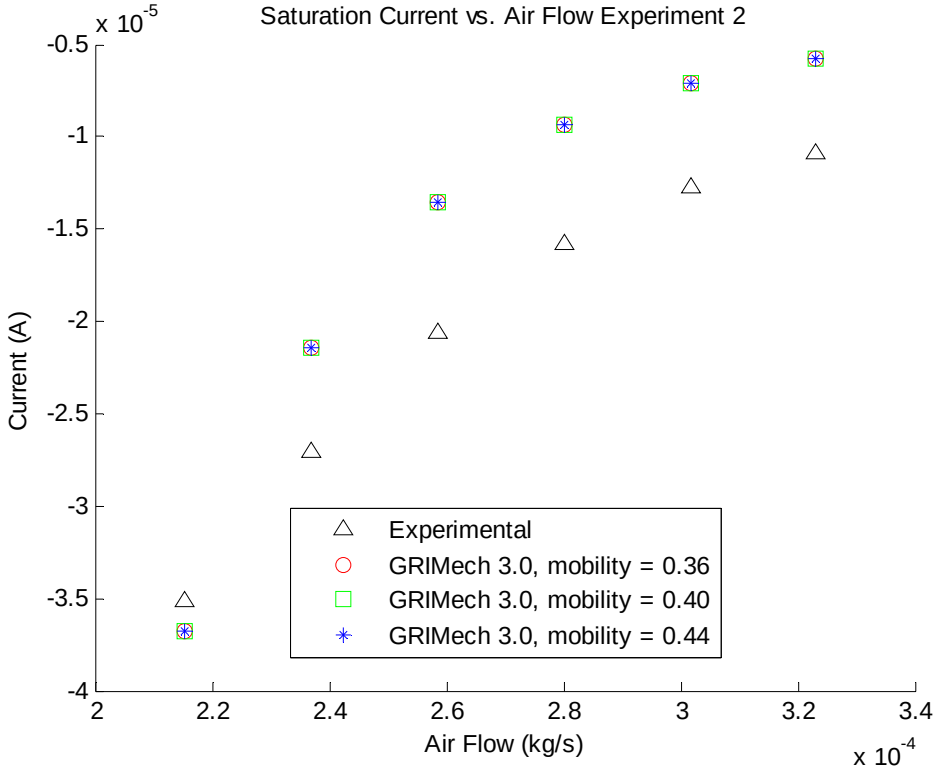

Figure 4-3: Saturation Currents Using GRIMech3.0 with Methane in Experiment 2

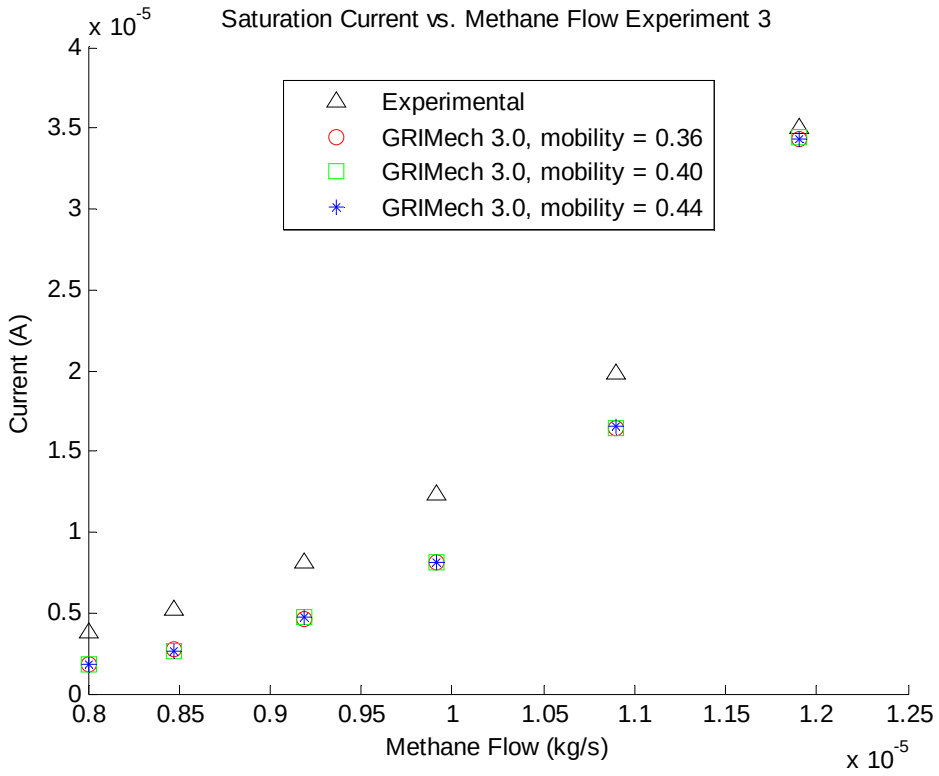

Figure 4-4: Saturation Currents Using GRIMech3.0 with Methane in Experiment 3

Although this mechanism overpredicted the conductivity values using the three tested mobility values, a significant improvement can be seen using the lowest of the tested mobility values, especially in experiments one and three which both had a burner to electrode distance of 1.6cm. Additionally, there appears to be decreased sensitivity to mobility changes as the air flow is increased. These effects can be seen in the following plots.

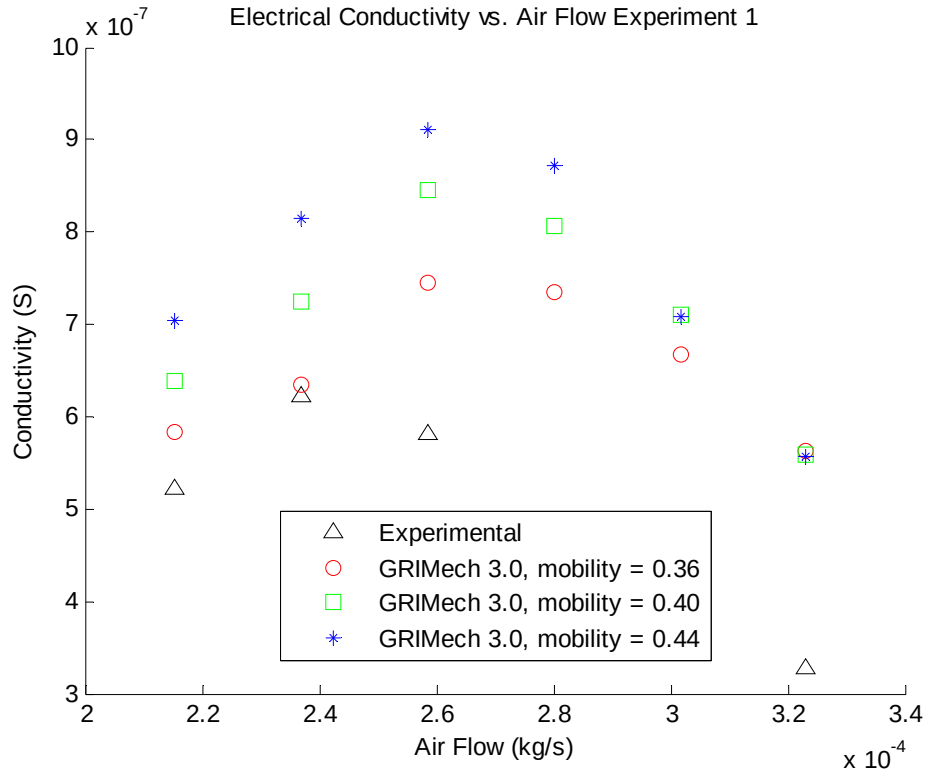

Figure 4-5: Conductivity Using GRIMech3.0 with Methane in Experiment 1

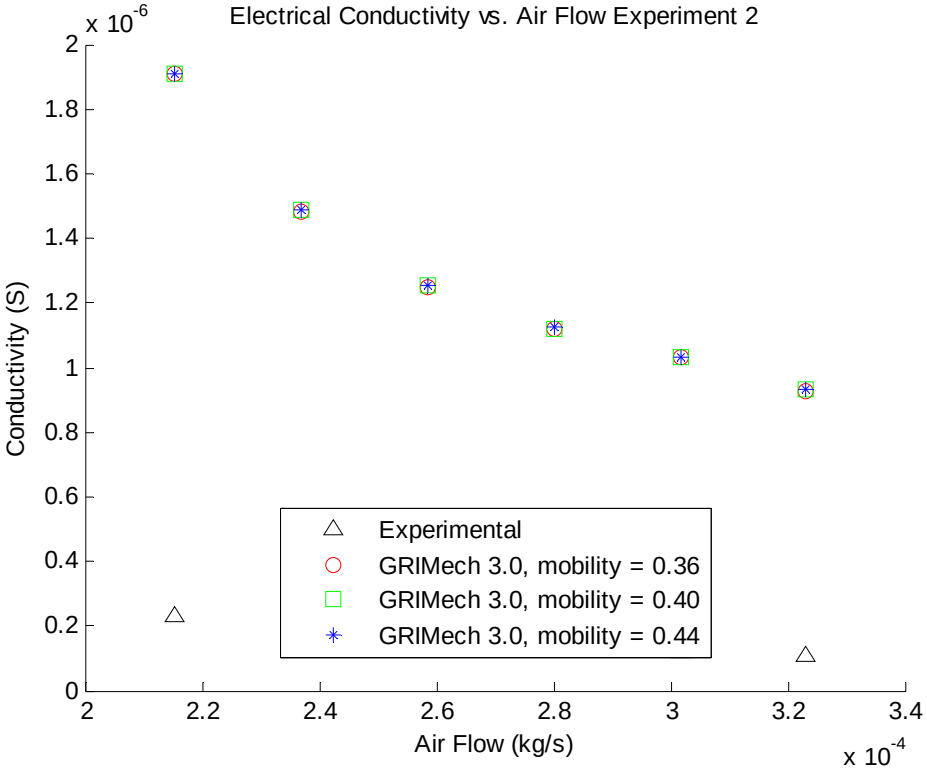

Figure 4-6: Conductivity Using GRIMech3.0 with Methane in Experiment 2

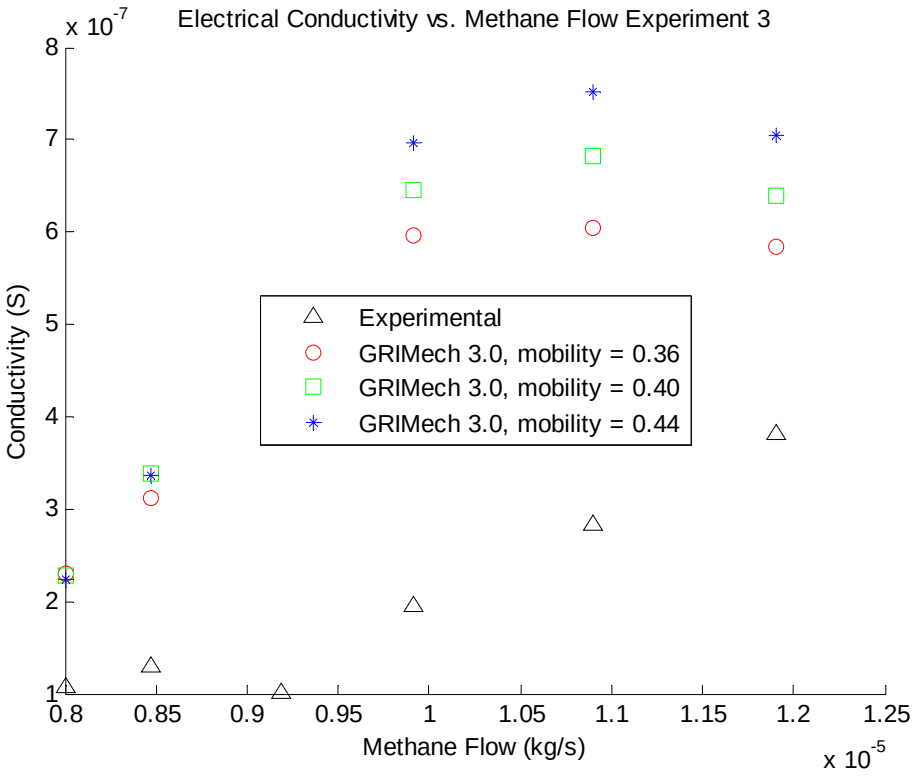

Figure 4-7: Conductivity Using GRIMech3.0 with Methane in Experiment 3

4.1.2 Jones, Becker and Heinsohn

The Jones, Becker and Heinsohn mechanism was one of the best two mechanism for predicting electrical characteristics of a burning methane flame. It had originally been designed for the purpose of simulation of electrical properties of flame.

Observation of the V-I curve plots in Appendix E reveals that the prediction of the saturation currents are reasonably close in most instances. The best agreement for the saturation current values can be observed towards the middle equivalence ratios in the tested range of 0.67 to 1.0, with underestimates approaching the stochiometric value of 1.0 and overpredictions approaching lean blow-off.

However, the conductivity, illustrated by the V-I curve slopes, remains noticably higher than experimentally shown with all tested mobility values. The conductivity error decreases with the electron mobility. This finding indicates that lowering the mobility may result in significantly better predictions of conductivity. The close predictions of saturation current together with excessive predictions for conductivity cause the saturation current to be reached at lower voltages than shown in the experimental data.

The following tables show properties of the V-I curves up to 90% of the saturation current for each of tested mobily value.

Experiment Number	Jones, Becker, and Heinsohn Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.36 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
2-1	1.0400E-06	8.0990E-07	-2.8200E-05	6.9100E-06	-38	121.96
2-2	8.9500E-07	7.2731E-07	-2.1500E-05	5.5770E-06	-35	124.95
2-3	8.1000E-07	6.6534E-07	-1.7400E-05	3.1800E-06	-32	107.95
2-4	7.5000E-07	6.2130E-07	-1.5000E-05	7.8100E-07	-30	89.95
2-5	7.0700E-07	5.8796E-07	-1.3000E-05	-3.0500E-07	-28	76.95
2-6	6.6500E-07	5.5790E-07	-1.1900E-05	-1.0110E-06	-27	72.949
3-2	5.7400E-07	2.9029E-07	1.6300E-05	-3.5590E-06	27	-42.943
3-3	5.7000E-07	3.7471E-07	1.0500E-05	-1.8340E-06	17	-42.947
3-4	6.2900E-07	5.2791E-07	7.5000E-06	-6.0200E-07	11	-38.945
3-5	7.0300E-07	5.7247E-07	5.3600E-06	9.9500E-08	7	-32.949
3-6	7.7500E-07	6.6696E-07	4.2800E-06	4.8150E-07	5	-29.946

Table 4-4: V-I Curve Characteristics to 90% of Maximum Current Using Jones, Becker, and Heinsohn with Electron Mobility of 0.36 m^2/Vs

Experiment Number	Jones, Becker, and Heinsohn Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.40 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	6.4100E-07	1.1948E-07	2.6600E-05	-4.9490E-06	40.00	-19.876
1-2	6.9700E-07	7.5310E-08	2.0500E-05	-4.2670E-06	28.00	-11.878
1-3	7.5600E-07	1.7467E-07	1.6800E-05	-5.8600E-07	21.00	-8.894
1-4	8.4000E-07	3.6203E-07	1.4400E-05	1.1900E-07	16.00	-13.894
1-5	9.8600E-07	5.9482E-07	1.2700E-05	1.0080E-06	12.00	-17.894
1-6	1.0800E-06	7.5312E-07	1.1600E-05	1.8542E-06	10.00	-19.894
2-1	1.0500E-06	8.1990E-07	-2.8200E-05	6.9100E-06	-38.00	121.96
2-2	9.0100E-07	7.3331E-07	-2.1500E-05	5.5770E-06	-35.00	124.95
2-3	8.1500E-07	6.7034E-07	-1.7400E-05	3.1800E-06	-32.00	107.95
2-4	7.5500E-07	6.2630E-07	-1.5000E-05	7.8100E-07	-30.00	89.95
2-5	7.1200E-07	5.9296E-07	-1.3000E-05	-3.0500E-07	-28.00	76.95
2-6	6.7000E-07	5.6290E-07	-1.1900E-05	-1.0110E-06	-27.00	72.949
3-1	6.4100E-07	2.5858E-07	2.6600E-05	-8.4780E-06	40.00	-49.953
3-2	6.6600E-07	3.8229E-07	1.6100E-05	-3.7590E-06	23.00	-46.943
3-3	6.4500E-07	4.4971E-07	1.0500E-05	-1.8340E-06	15.00	-44.947
3-4	6.9000E-07	5.8891E-07	7.5200E-06	-5.8200E-07	10.00	-39.945
3-5	8.2700E-07	6.9647E-07	5.4400E-06	1.7950E-07	6.00	-33.949
3-6	9.5200E-07	8.4396E-07	4.2400E-06	4.4150E-07	4.00	-30.946

Table 4-5: V-I Curve Characteristics to 90% of Maximum Current Using Jones, Becker, and Heinsohn with Electron Mobility of 0.40 m^2/Vs

Experiment Number	Jones, Becker, and Heinsohn Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.44 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	7.1100E-07	1.8948E-07	2.6700E-05	-4.8490E-06	36.00	-23.876
1-2	7.7900E-07	1.5731E-07	2.0600E-05	-4.1670E-06	25.00	-14.878
1-3	8.7200E-07	2.9067E-07	1.6700E-05	-6.8600E-07	18.00	-11.894
1-4	9.6900E-07	4.9103E-07	1.4600E-05	3.1900E-07	14.00	-15.894
1-5	1.1700E-06	7.7882E-07	1.2700E-05	1.0080E-06	10.00	-19.894
1-6	1.3300E-06	1.0031E-06	1.1500E-05	1.7542E-06	8.00	-21.894
2-1	1.0500E-06	8.1990E-07	-2.8100E-05	7.0100E-06	-38.00	121.96
2-2	9.0500E-07	7.3731E-07	-2.1500E-05	5.5770E-06	-35.00	124.95
2-3	8.1700E-07	6.7234E-07	-1.7400E-05	3.1800E-06	-32.00	107.95
2-4	7.6000E-07	6.3130E-07	-1.5000E-05	7.8100E-07	-30.00	89.95
2-5	7.1500E-07	5.9596E-07	-1.3000E-05	-3.0500E-07	-28.00	76.95
2-6	6.7400E-07	5.6690E-07	-1.1800E-05	-9.1100E-07	-27.00	72.949
3-1	7.1100E-07	3.2858E-07	2.6700E-05	-8.3780E-06	36.00	-53.953
3-2	7.3400E-07	4.5029E-07	1.6300E-05	-3.5590E-06	21.00	-48.943
3-3	7.4800E-07	5.5271E-07	1.0500E-05	-1.8340E-06	13.00	-46.947
3-4	7.7500E-07	6.7391E-07	7.5800E-06	-5.2200E-07	9.00	-40.945
3-5	9.8400E-07	8.5347E-07	5.4300E-06	1.6950E-07	5.00	-34.949
3-6	9.6800E-07	8.5996E-07	4.3400E-06	5.4150E-07	4.00	-30.946

Table 4-6: V-I Curve Characteristics to 90% of Maximum Current Using Jones, Becker, and Heinsohn with Electron Mobility of 0.40 m^2/Vs

The values of for the saturation currents found by this mechanism were nearly the same for all all tested mobility values, which is shown in the following figures, and in most cases, were in good agreement with the experimental values. This fact can be seen in the following figures.

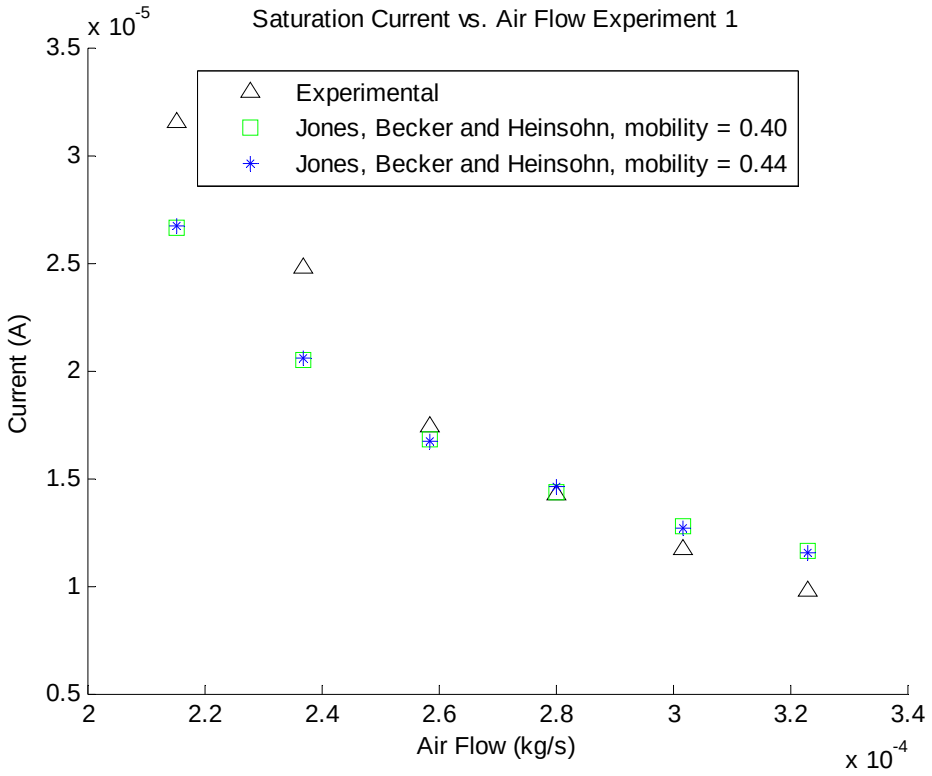

Figure 4-8: Saturation Current Using Jone, Becker and Heinsohn with Methane in Experiment 1

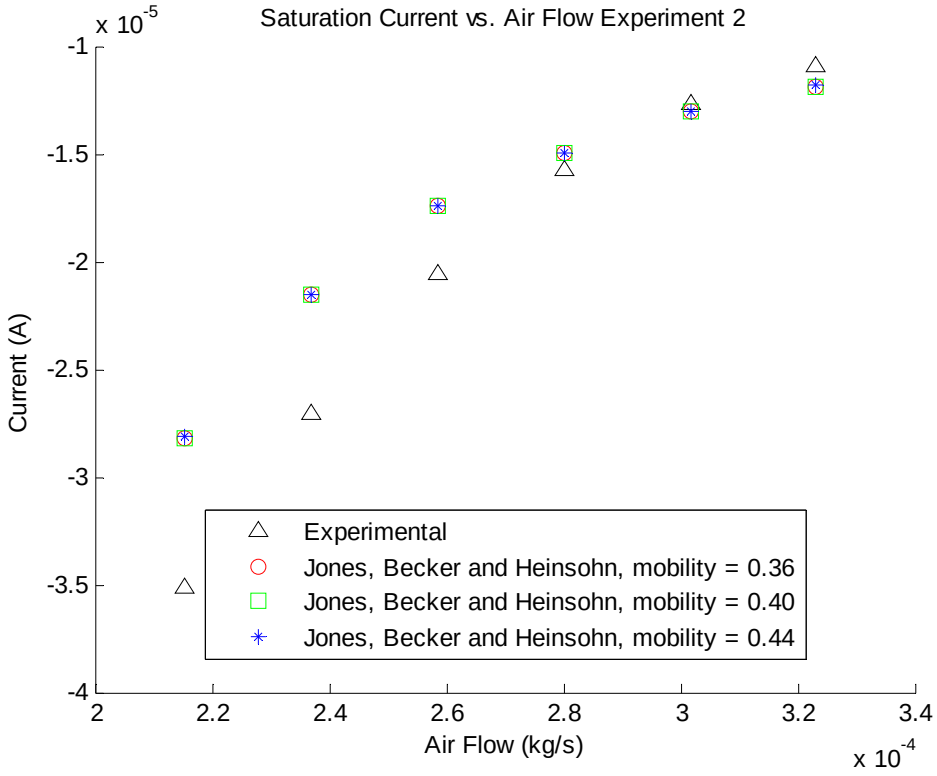

Figure 4-9: Saturation Current Using Jone, Becker and Heinsohn with Methane in Experiment 2

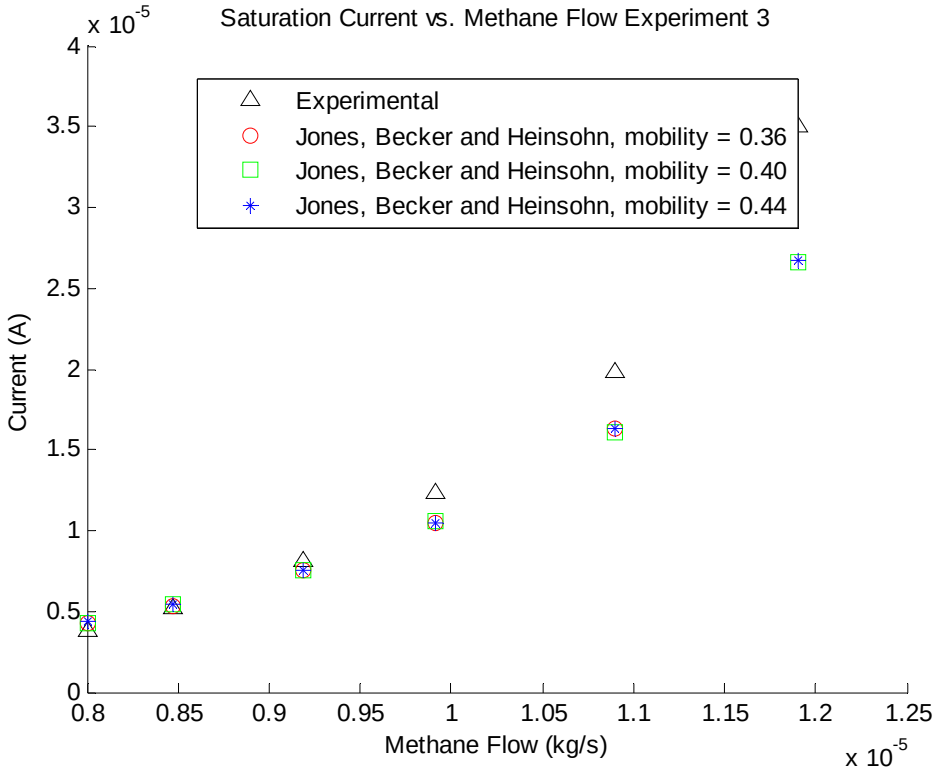

Figure 4-10: Saturation Current Using Jone, Becker and Heinsohn with Methane in Experiment 3

In all tested cases, the simulated conductivity is higher than the experimental conductivity, but there is a significant drop in simulated conductivity as the mobility is lowered. This effect is evident in the following figures.

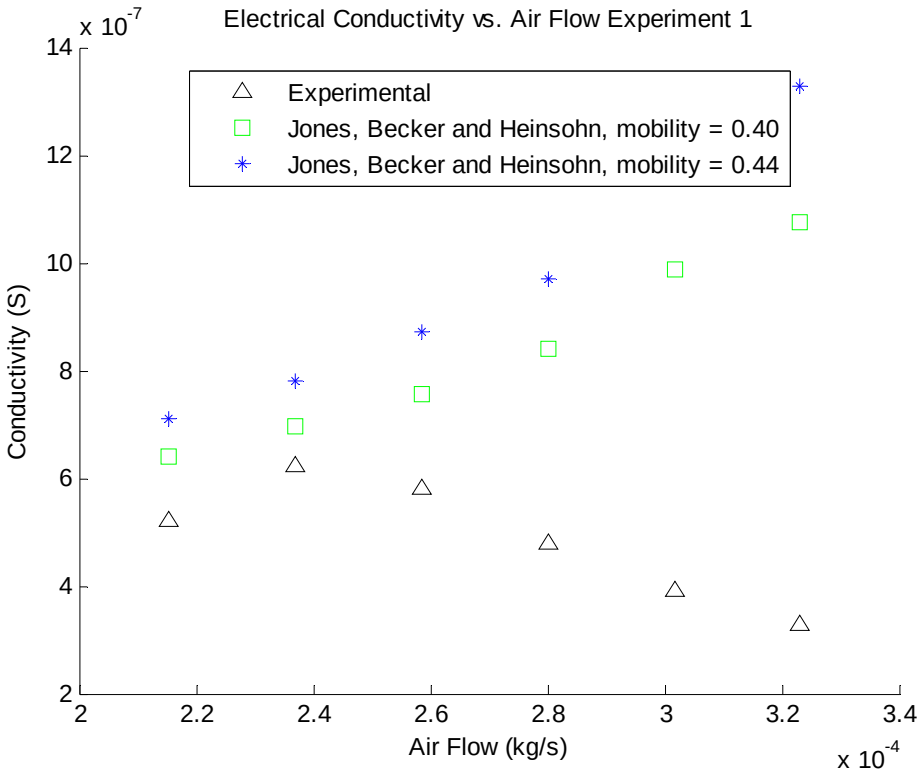

Figure 4-11: Conductivity Using Jone, Becker and Heinsohn with Methane in Experiment 1

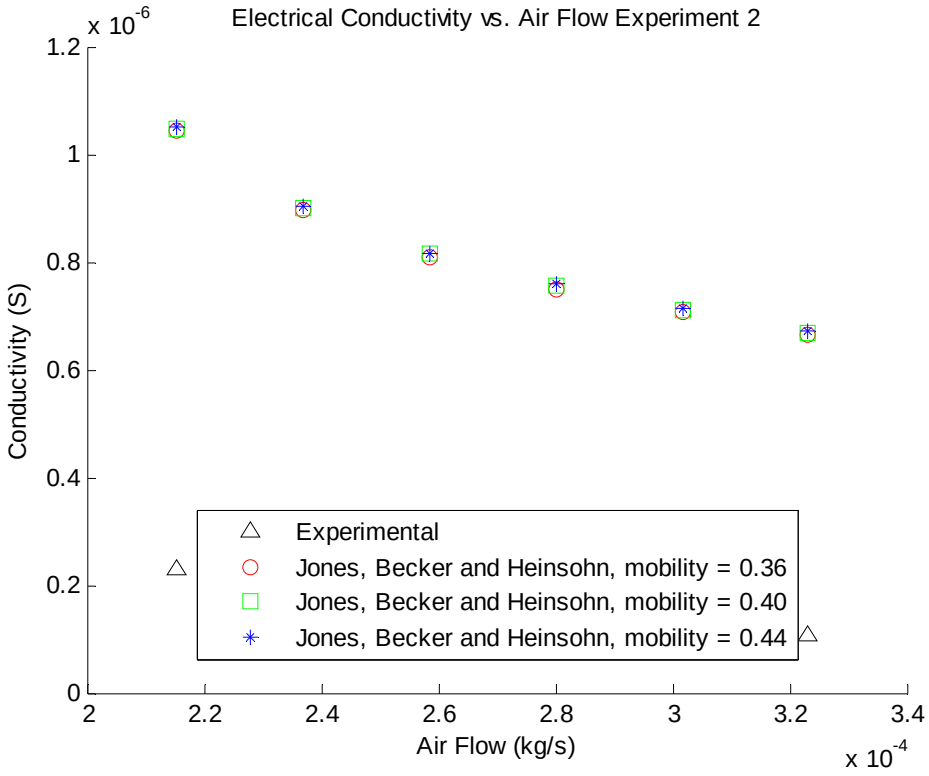

Figure 4-12: Conductivity Using Jone, Becker and Heinsohn with Methane in Experiment 2

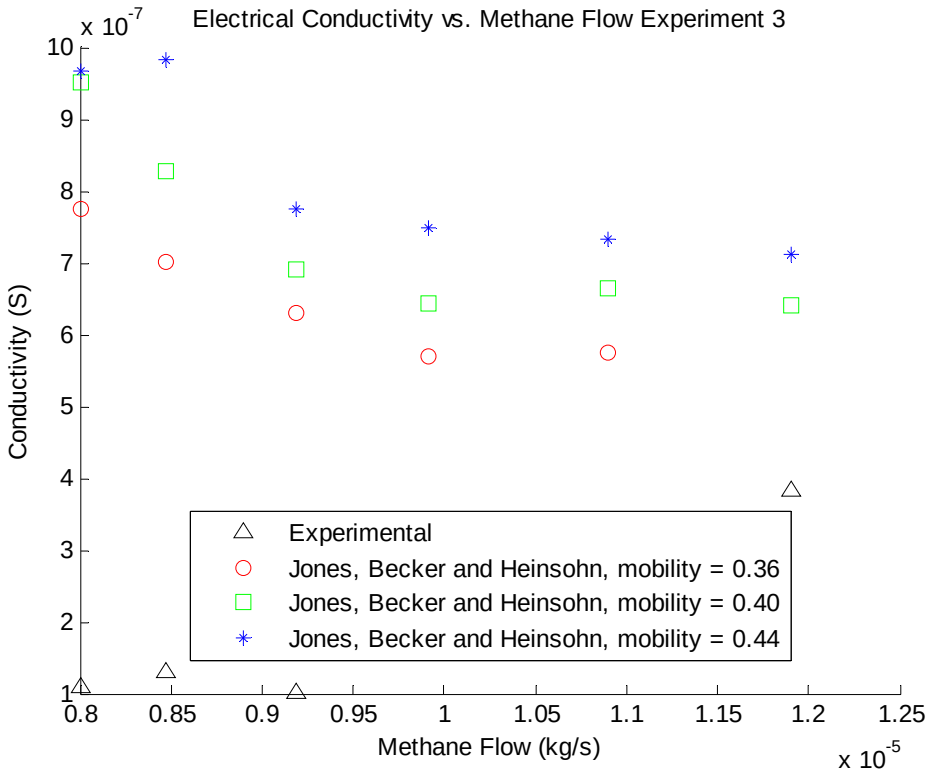

Figure 4-13: Conductivity Using Jone, Becker and Heinsohn with Methane in Experiment 3

4.1.3 Pederson and Brown

In this research, the Pederson and Brown mechanism was the worst performing mechanism out of the four. Very little charged particle concentrations were predicted, resulting in extremely low saturation currents, and unreliable predictions for conductivity. One potential source of errors may be in the use of a scanner and optical character recognition software for retrieving the reaction rate values. In addition, this mechanism used a different set of charged particle reactions than the other three mechanisms.

The following tables show properties of the V-I curves generated by this mechanism based on existing experiments up to 90% of the saturation current for each of tested mobily values.

Experiment Number	Pederson and Brown Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.36 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	1.0100E-06	4.8848E-07	2.6900E-06	-2.8859E-05	2.00	-57.876
1-2	3.0900E-07	-3.1269E-07	4.5500E-07	-2.4312E-05	1.00	-38.878
1-3	1.1500E-07	-4.6633E-07	1.6900E-07	-1.7217E-05	1.00	-28.894
1-4	3.3400E-08	-4.4457E-07	9.3100E-08	-1.4188E-05	2.00	-27.894
1-5	2.0700E-08	-3.7048E-07	5.7100E-08	-1.1635E-05	2.00	-27.894
1-6	0.0000E+00	-3.2688E-07	0.0000E+00	-9.7458E-06	0.00	-29.894
2-1	3.9700E-07	1.6690E-07	-2.6500E-06	3.2460E-05	-11.00	148.96
2-2	1.4600E-07	-2.1690E-08	-4.7900E-07	2.6598E-05	-5.00	154.95
2-3	8.2700E-08	-6.1960E-08	-1.7600E-07	2.0404E-05	-3.00	136.95
2-4	4.1900E-08	-8.6800E-08	-9.7200E-08	1.5684E-05	-3.00	116.95
2-5	3.5900E-08	-8.3140E-08	-5.9000E-08	1.2636E-05	-2.00	102.95
2-6	2.3700E-08	-8.3400E-08	-4.1600E-08	1.0847E-05	-2.00	97.949
3-1	1.0100E-06	6.2758E-07	2.6900E-06	-3.2388E-05	2.00	-87.953
3-2	2.4600E-07	-3.7710E-08	3.6200E-07	-1.9497E-05	1.00	-68.943
3-3	6.6100E-08	-1.2919E-07	9.6400E-08	-1.2238E-05	1.00	-58.947
3-4	1.5600E-08	-8.5490E-08	4.3300E-08	-8.0587E-06	2.00	-47.945
3-5	0.0000E+00	-1.3053E-07	0.0000E+00	-5.2605E-06	0.00	-39.949
3-6	4.2700E-09	-1.0377E-07	1.1500E-08	-3.7870E-06	2.00	-32.946

Table 4-7: V-I Curve Characteristics to 90% of Maximum Current Using Pederson and Brown with Electron Mobility of 0.36 m^2/Vs

Experiment Number	Pederson and Brown Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.40 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	1.0000E-06	4.7848E-07	2.7300E-06	-2.8819E-05	2.00	-57.876
1-2	3.0900E-07	-3.1269E-07	4.5800E-07	-2.4309E-05	1.00	-38.878
1-3	1.1500E-07	-4.6633E-07	1.7000E-07	-1.7216E-05	1.00	-28.894
1-4	3.3100E-08	-4.4487E-07	9.3300E-08	-1.4188E-05	2.00	-27.894
1-5	2.0500E-08	-3.7068E-07	5.7200E-08	-1.1635E-05	2.00	-27.894
1-6	1.4200E-08	-3.1268E-07	3.9200E-08	-9.7066E-06	2.00	-27.894
2-1	4.0000E-07	1.6990E-07	-2.6500E-06	3.2460E-05	-11.00	148.96
2-2	1.4800E-07	-1.9690E-08	-4.7900E-07	2.6598E-05	-5.00	154.95
2-3	8.3100E-08	-6.1560E-08	-1.7600E-07	2.0404E-05	-3.00	136.95
2-4	5.1300E-08	-7.7400E-08	-7.5400E-08	1.5706E-05	-2.00	117.95
2-5	3.1800E-08	-8.7240E-08	-5.1000E-08	1.2644E-05	-2.00	102.95
2-6	2.1300E-08	-8.5800E-08	-3.7000E-08	1.0852E-05	-2.00	97.949
3-1	1.0000E-06	6.1758E-07	2.7300E-06	-3.2348E-05	2.00	-87.953
3-2	2.4700E-07	-3.6710E-08	3.6400E-07	-1.9495E-05	1.00	-68.943
3-3	6.5700E-08	-1.2959E-07	9.7000E-08	-1.2237E-05	1.00	-58.947
3-4	1.5400E-08	-8.5690E-08	4.3400E-08	-8.0586E-06	2.00	-47.945
3-5	7.2900E-09	-1.2324E-07	1.9400E-08	-5.2411E-06	2.00	-37.949
3-6	4.2400E-09	-1.0380E-07	1.1500E-08	-3.7870E-06	2.00	-32.946

Table 4-8: V-I Curve Characteristics to 90% of Maximum Current Using Pederson and Brown with Electron Mobility of 0.40 m^2/Vs

Experiment Number	Pederson and Brown Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.44 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	9.7989E-07	4.5837E-07	2.7200E-06	-2.8829E-05	2.00	-57.876
1-2	3.0639E-07	-3.1530E-07	4.6119E-07	-2.4306E-05	1.00	-38.878
1-3	1.1387E-07	-4.6746E-07	1.7094E-07	-1.7215E-05	1.00	-28.894
1-4	3.2769E-08	-4.4520E-07	9.3451E-08	-1.4188E-05	2.00	-27.894
1-5	2.0383E-08	-3.7080E-07	5.7260E-08	-1.1635E-05	2.00	-27.894
2-1	4.0197E-07	1.7187E-07	-2.6490E-06	3.2461E-05	-11.00	148.96
2-2	1.4850E-07	-1.9190E-08	-4.7892E-07	2.6598E-05	-5.00	154.95
2-3	8.3474E-08	-6.1186E-08	-1.7627E-07	2.0404E-05	-3.00	136.95
2-4	4.2126E-08	-8.6574E-08	-9.7240E-08	1.5684E-05	-3.00	116.95
2-5	3.6003E-08	-8.3037E-08	-5.8985E-08	1.2636E-05	-2.00	102.95
2-6	2.3786E-08	-8.3314E-08	-4.1633E-08	1.0847E-05	-2.00	97.949
3-1	9.7989E-07	5.9747E-07	2.7200E-06	-3.2358E-05	2.00	-87.953
3-2	2.4178E-07	-4.1930E-08	3.6619E-07	-1.9493E-05	1.00	-68.943
3-3	6.5535E-08	-1.2976E-07	9.7825E-08	-1.2236E-05	1.00	-58.947
3-4	2.8132E-08	-7.2958E-08	4.1054E-08	-8.0609E-06	1.00	-48.945
3-5	6.8537E-09	-1.2368E-07	1.9445E-08	-5.2411E-06	2.00	-37.949
3-6	4.2071E-09	-1.0383E-07	1.1560E-08	-3.7869E-06	2.00	-32.946

Table 4-9: V-I Curve Characteristics to 90% of Maximum Current Using Pederson and Brown with Electron Mobility of 0.44 m^2/Vs

The following figures show that for each of the tested mobilities, the saturation current is underpredicted by one or more orders of magnitude.

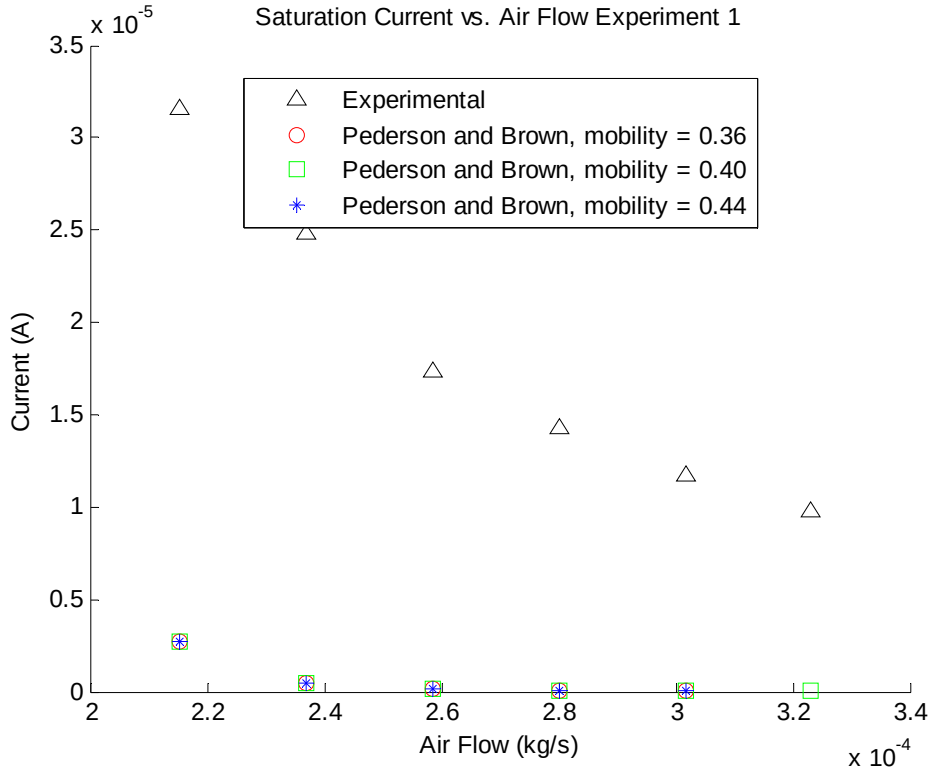

Figure 4-14: Saturation Current Using Pederson and Brown with Methane in Experiment 1

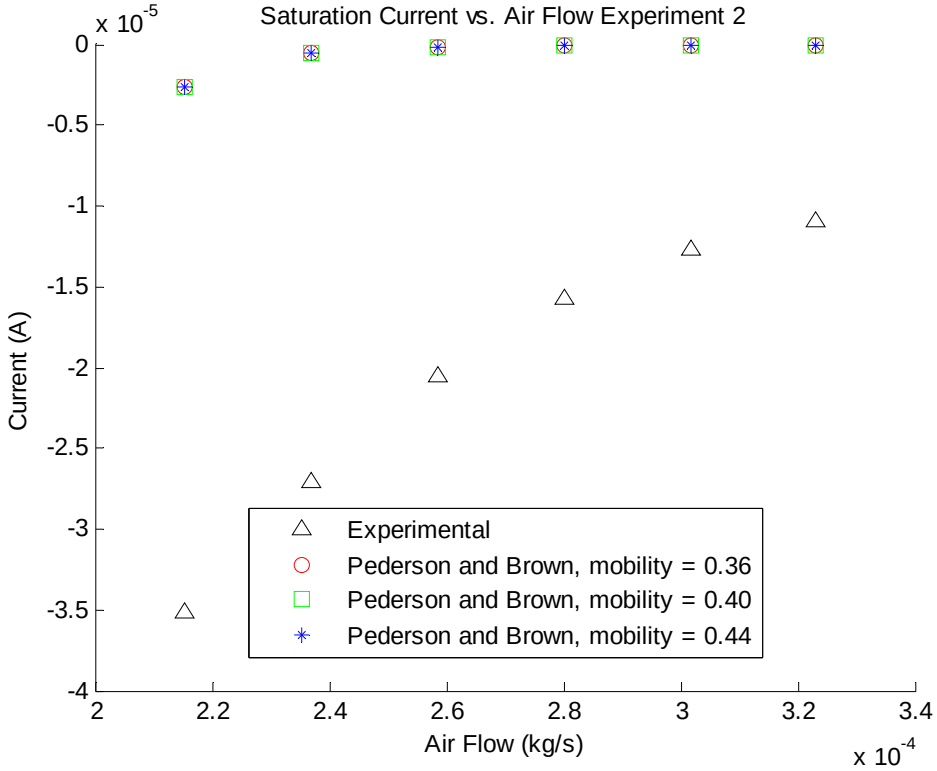

Figure 4-15: Saturation Current Using Pederson and Brown with Methane in Experiment 2

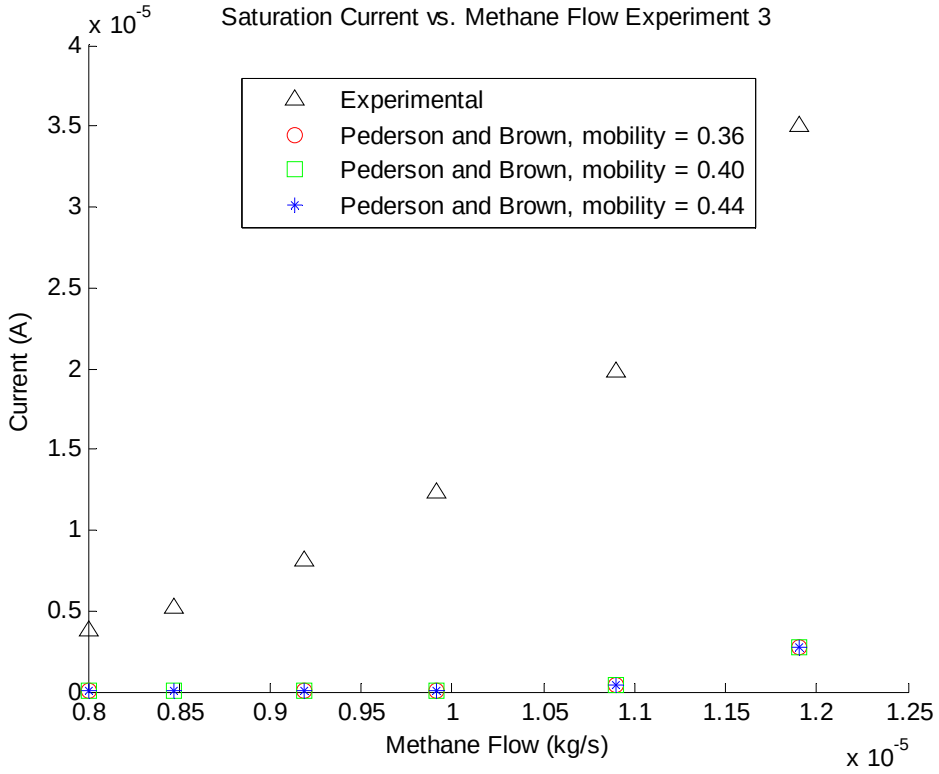

Figure 4-16: Saturation Current Using Pederson and Brown with Methane in Experiment 3

In nearly all of the simulated experiments, the conductivity for this mechanism actually underpredicted the experimental values, although there are a few exceptions as seen in the following figures. However, the problems with the low charged particle concentration and saturation currents makes these conductivity predictions unreliable.

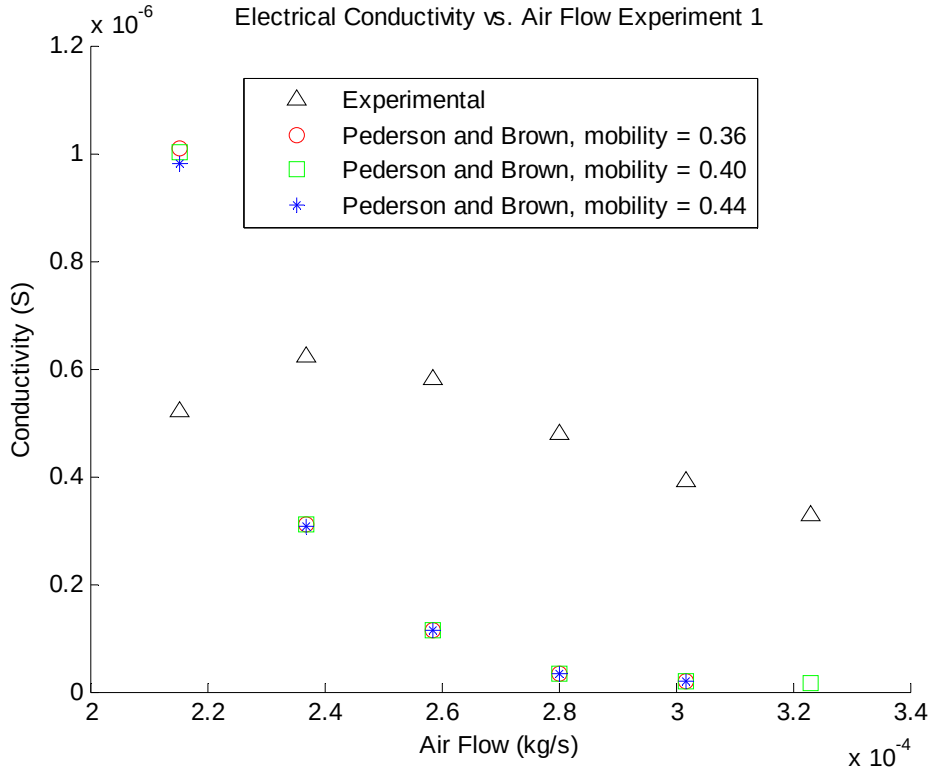

Figure 4-17: Conductivity Using Pederson and Brown with Methane in Experiment 1

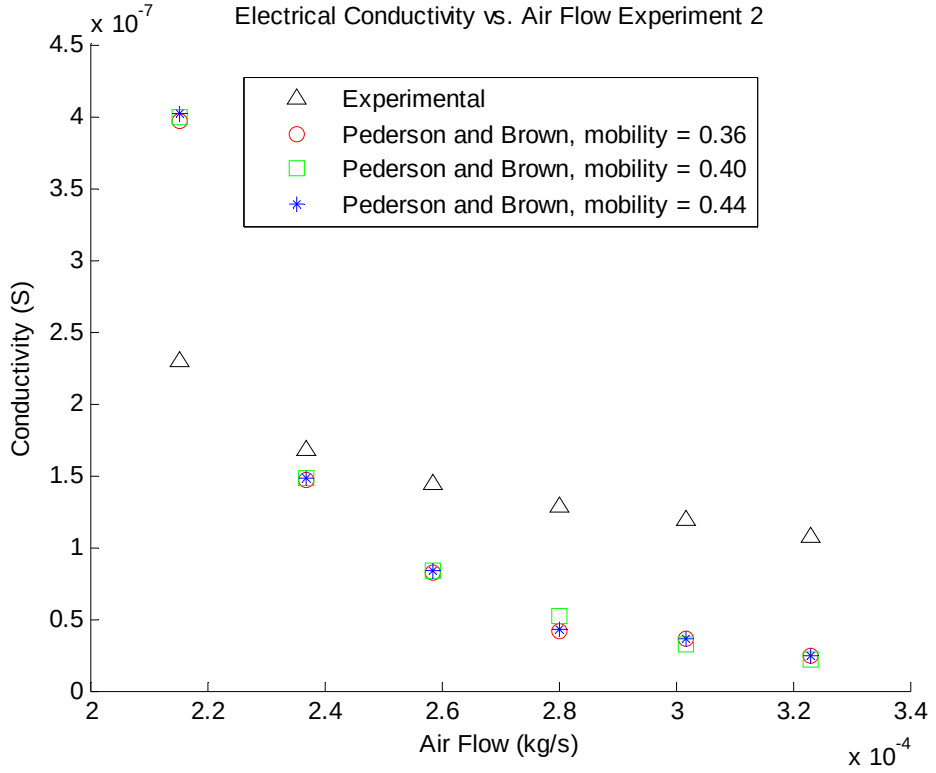

Figure 4-18: Conductivity Using Pederson and Brown with Methane in Experiment 2

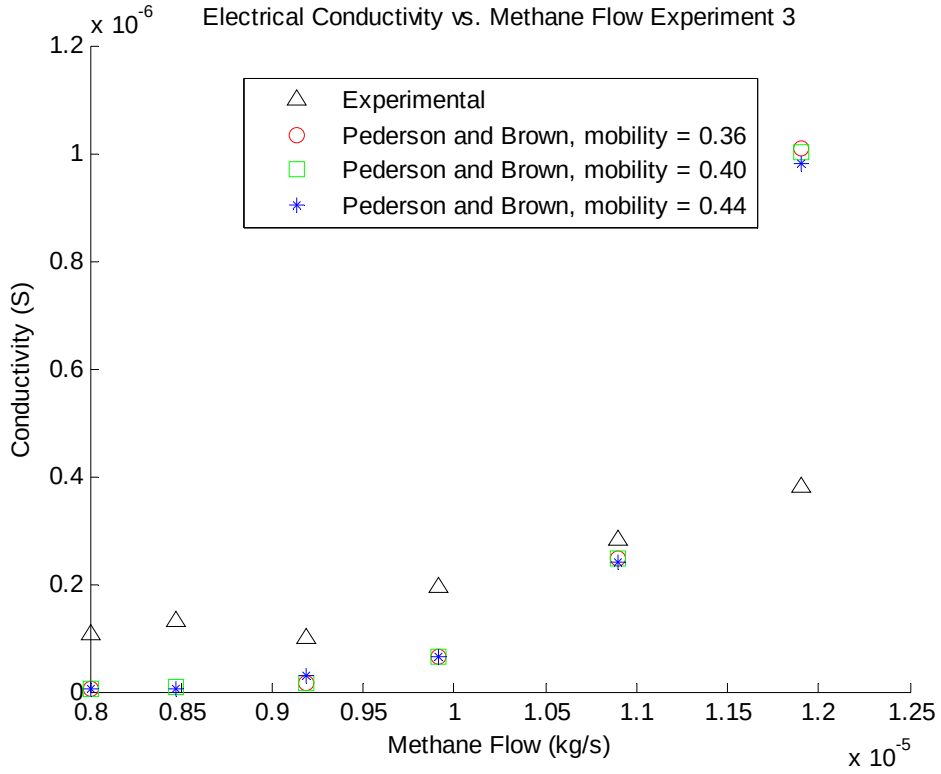

Figure 4-19: Conductivity Using Pederson and Brown with Methane in Experiment 3

4.1.4 Peters

The modified Peters mechanism did not perform as well as the Jones, Becker and Heinsohn or GRIMech 3.0. However it did have some success in predicting general trends for the electrical properties of natural gas flames.

The V-I curve plots in Appendix E quickly reveal that the prediction of the saturation currents are significantly too high in all instances. The actual value of the error appears to be lower for the most lean experiments.

The conductivity, illustrated by the V-I curve slopes, is significantly higher than the conductivity shown in the lab tests. The error with conductivity decreases with the electron mobility.

The following tables show properties of the V-I curves up to 90% of the saturation current for each of the tested mobily values.

Experiment Number	Pederson and Brown Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.36 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	9.3342E-07	4.1190E-07	1.5457E-04	1.2302E-04	163.00	103.124
1-2	8.1003E-07	1.8834E-07	8.8866E-05	6.4099E-05	107.00	67.122
1-3	8.1107E-07	2.2974E-07	5.6366E-05	3.8980E-05	67.00	37.106
1-4	7.7691E-07	2.9894E-07	3.9164E-05	2.4883E-05	48.00	18.106
1-5	8.8662E-07	4.9544E-07	2.9277E-05	1.7585E-05	31.00	1.106
1-6	9.1855E-07	5.9167E-07	2.2957E-05	1.3211E-05	23.00	-6.894
2-1	4.0648E-06	3.8347E-06	-1.6317E-04	-1.2806E-04	-43.00	116.96
2-2	2.9359E-06	2.7682E-06	-9.3014E-05	-6.5937E-05	-35.00	124.95
2-3	2.3669E-06	2.2222E-06	-5.8580E-05	-3.8000E-05	-28.00	111.95
2-4	2.0791E-06	1.9504E-06	-3.9754E-05	-2.3973E-05	-22.00	97.95
2-5	1.9030E-06	1.7840E-06	-2.9630E-05	-1.6935E-05	-18.00	86.95
3-1	9.3342E-07	5.5100E-07	1.5457E-04	1.1949E-04	163.00	73.047
3-2	7.4680E-07	4.6309E-07	7.2927E-05	5.3068E-05	95.00	25.057
3-3	6.9421E-07	4.9892E-07	3.5616E-05	2.3282E-05	49.00	-10.947
3-4	6.8047E-07	5.7938E-07	2.0372E-05	1.2270E-05	28.00	-21.945
3-5	5.8794E-07	4.5741E-07	1.1656E-05	6.3955E-06	18.00	-21.949
3-6	4.7814E-07	3.7010E-07	8.1009E-06	4.3024E-06	15.00	-19.946

Table 4-10: V-I Curve Characteristics to 90% of Maximum Current Using Peters with Electron Mobility of 0.36 m^2/Vs

Experiment Number	Pederson and Brown Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.40 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	1.0023E-06	4.8078E-07	1.5499E-04	1.2344E-04	152.00	92.124
1-2	8.8115E-07	2.5946E-07	8.8739E-05	6.3972E-05	98.00	58.122
1-3	9.0168E-07	3.2035E-07	5.6301E-05	3.8915E-05	60.00	30.106
1-4	8.6687E-07	3.8890E-07	3.9306E-05	2.5025E-05	43.00	13.106
1-5	9.7720E-07	5.8602E-07	2.9324E-05	1.7632E-05	28.00	-1.894
1-6	1.0556E-06	7.2872E-07	2.3079E-05	1.3333E-05	20.00	-9.894
2-1	4.0674E-06	3.8373E-06	-1.6314E-04	-1.2803E-04	-43.00	116.96
2-2	2.9386E-06	2.7709E-06	-9.2998E-05	-6.5921E-05	-35.00	124.95
2-3	2.3696E-06	2.2249E-06	-5.8568E-05	-3.7988E-05	-28.00	111.95
2-4	2.0819E-06	1.9532E-06	-3.9746E-05	-2.3965E-05	-22.00	97.95
2-5	1.9056E-06	1.7866E-06	-2.9625E-05	-1.6930E-05	-18.00	86.95
3-1	1.0023E-06	6.1988E-07	1.5499E-04	1.1991E-04	152.00	62.047
3-2	8.1140E-07	5.2769E-07	7.2725E-05	5.2866E-05	87.00	17.057
3-3	7.6660E-07	5.7131E-07	3.5466E-05	2.3132E-05	44.00	-15.947
3-4	7.3403E-07	6.3294E-07	2.0511E-05	1.2409E-05	26.00	-23.945
3-5	6.1981E-07	4.8928E-07	1.1692E-05	6.4315E-06	17.00	-22.949
3-6	5.0226E-07	3.9422E-07	8.0254E-06	4.2269E-06	14.00	-20.946

Table 4-11: V-I Curve Characteristics to 90% of Maximum Current Using Peters with Electron Mobility of 0.40 m^2/Vs

Experiment Number	Pederson and Brown Methane Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.44 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1-1	1.0711E-06	5.4958E-07	1.5494E-04	1.2339E-04	142.00	82.124
1-2	9.4779E-07	3.2610E-07	8.8823E-05	6.4056E-05	91.00	51.122
1-3	9.9576E-07	4.1443E-07	5.6141E-05	3.8755E-05	54.00	24.106
1-4	1.0306E-06	5.5263E-07	3.9286E-05	2.5005E-05	36.00	6.106
1-5	1.0522E-06	6.6102E-07	2.9450E-05	1.7758E-05	26.00	-3.894
1-6	1.1011E-06	7.7422E-07	2.3017E-05	1.3271E-05	19.00	-10.894
2-1	4.0697E-06	3.8396E-06	-1.6313E-04	-1.2802E-04	-43.00	116.96
2-2	2.9410E-06	2.7733E-06	-9.2985E-05	-6.5908E-05	-35.00	124.95
2-3	2.3695E-06	2.2248E-06	-5.8512E-05	-3.7932E-05	-28.00	111.95
2-4	2.0831E-06	1.9544E-06	-3.9719E-05	-2.3938E-05	-22.00	97.95
2-5	1.9050E-06	1.7860E-06	-2.9592E-05	-1.6897E-05	-18.00	86.95
2-6	1.8055E-06	1.6984E-06	-2.3553E-05	-1.2664E-05	-15.00	84.949
3-1	1.0711E-06	6.8868E-07	1.5494E-04	1.1986E-04	142.00	52.047
3-2	9.1718E-07	6.3347E-07	7.2924E-05	5.3065E-05	77.00	7.057
3-3	8.2343E-07	6.2814E-07	3.5627E-05	2.3293E-05	41.00	-18.947
3-4	7.9098E-07	6.8989E-07	2.0514E-05	1.2412E-05	24.00	-25.945
3-5	6.5153E-07	5.2100E-07	1.1656E-05	6.3955E-06	16.00	-23.949
3-6	5.0527E-07	3.9723E-07	8.1271E-06	4.3286E-06	14.00	-20.946

Table 4-12: V-I Curve Characteristics to 90% of Maximum Current Using Peters with Electron Mobility of 0.44 m^2/Vs

The following figures show that the saturation current is overpredicted in all cases with this mechanism, although the values are closer in cases with lower equivalence ratios.

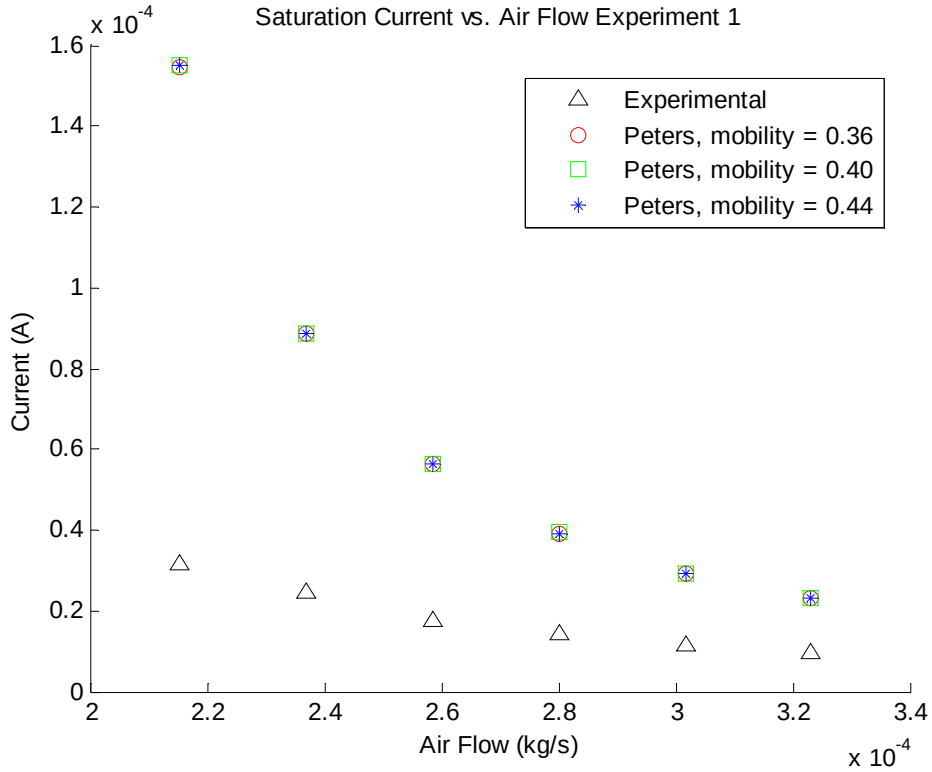

Figure 4-20: Saturation Current Using Peters with Methane in Experiment 1

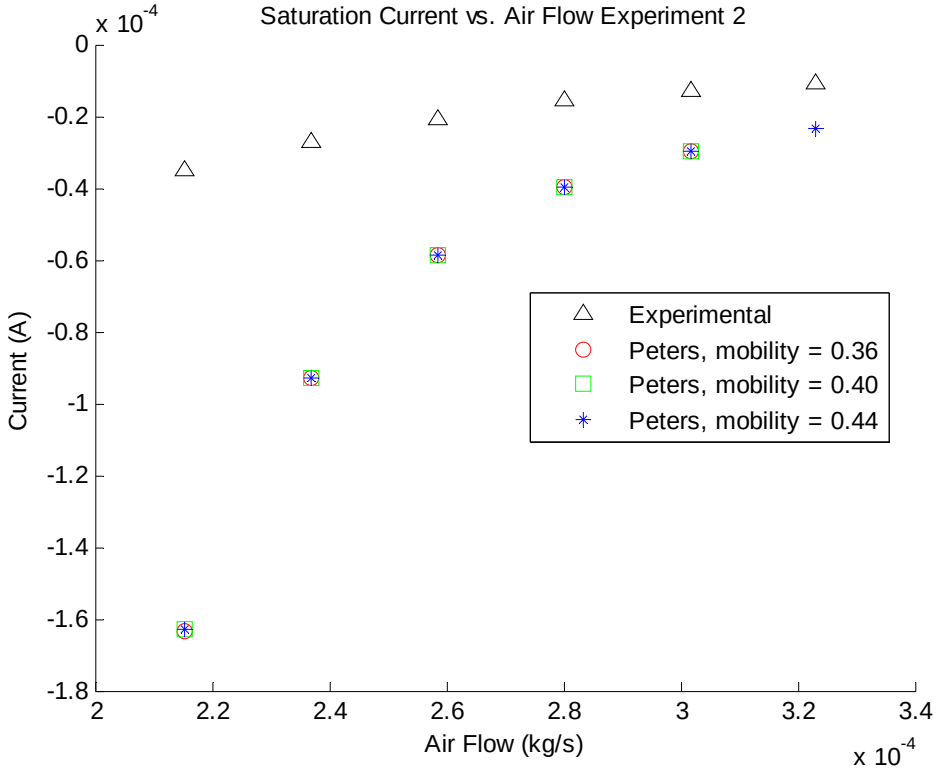

Figure 4-21: Saturation Current Using Peters with Methane in Experiment 2

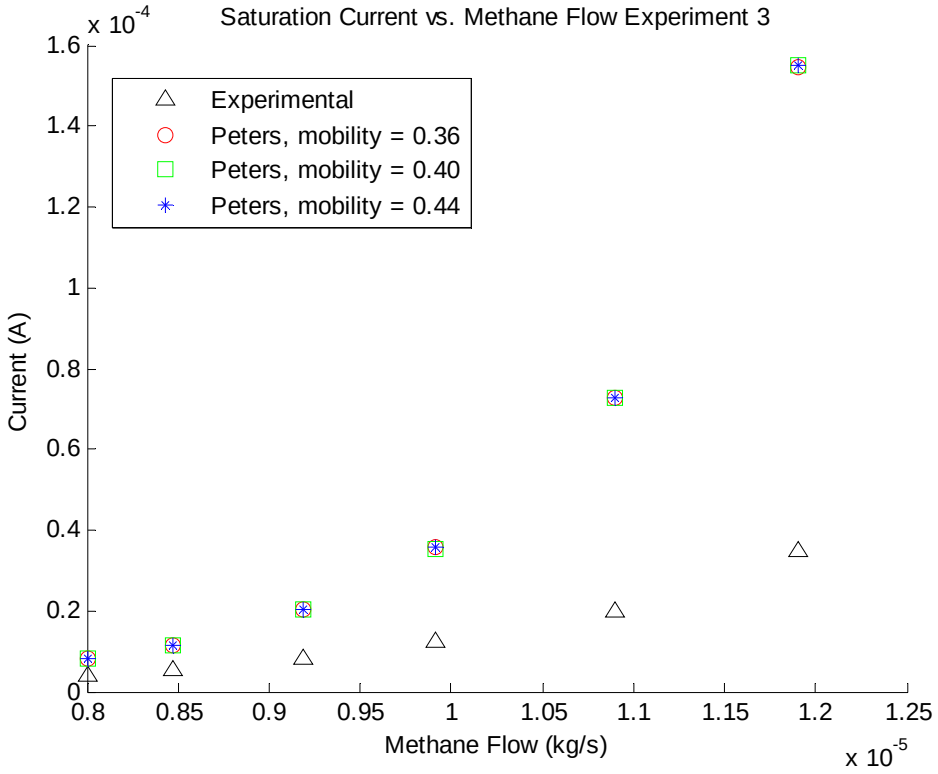

Figure 4-22: Saturation Current Using Peters with Methane in Experiment 3

The simulated conductivity using the tested mobilities is significantly higher than the experimental values for all of the natural gas simulations. For experiments one and three, which had a burner to electrode distance of 1.6cm, there was a noticeable reduction of the simulated conductivity with decreased mobility values, but for experiment 2, which had a burner to electrode distance of just 0.2cm, this effect was not noticeable.

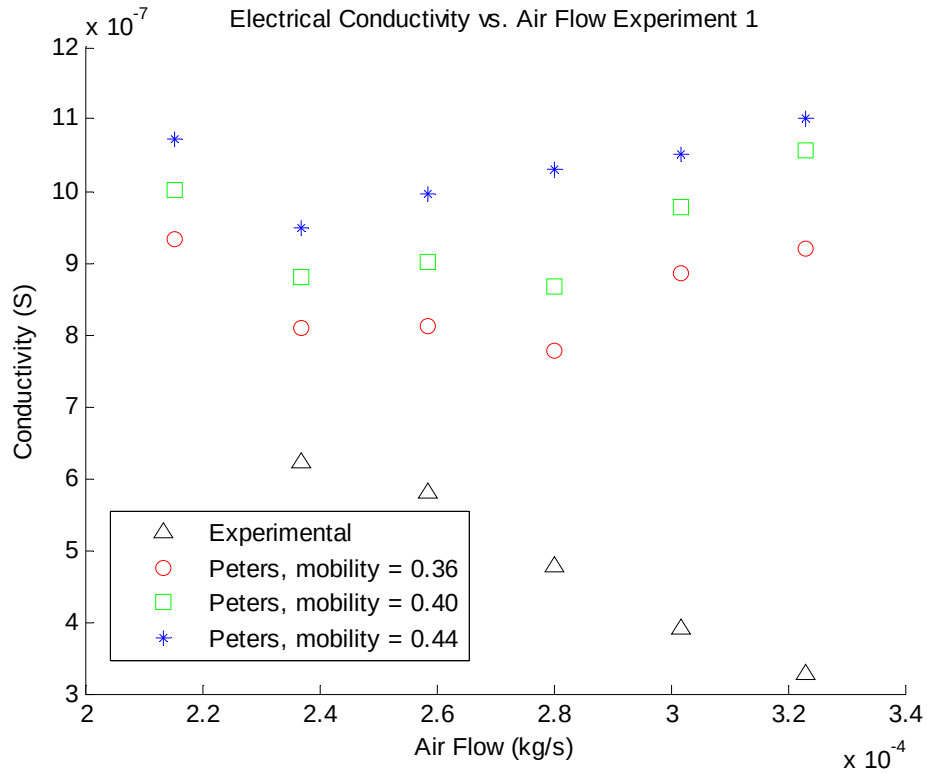

Figure 4-23: Conductivity Using Peters with Methane in Experiment 1

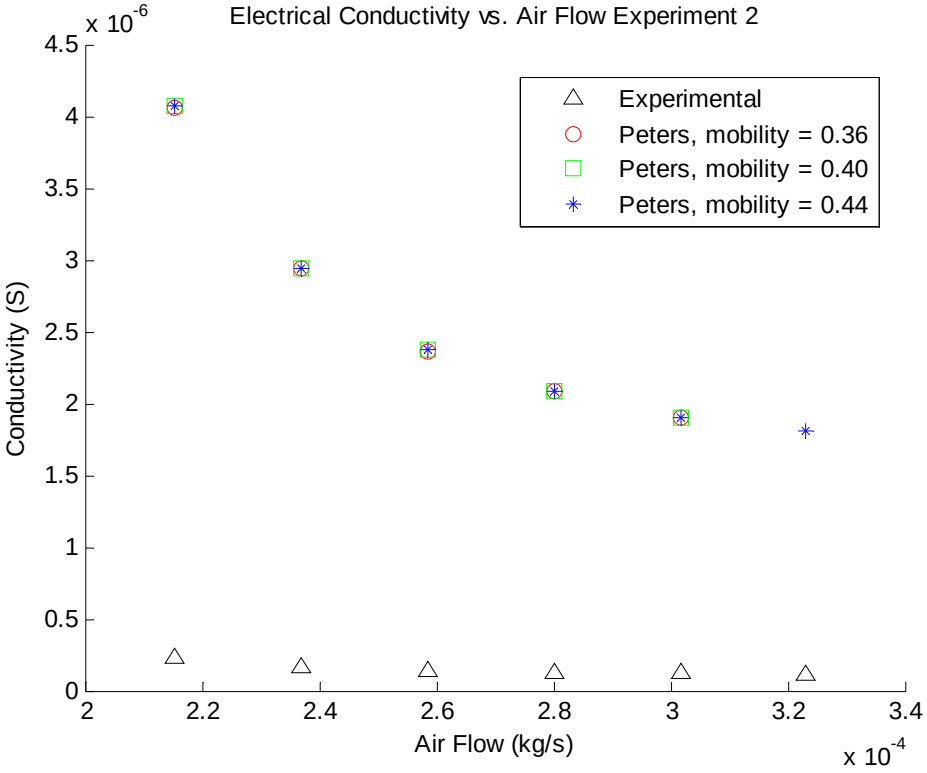

Figure 4-24: Conductivity Using Peters with Methane in Experiment 2

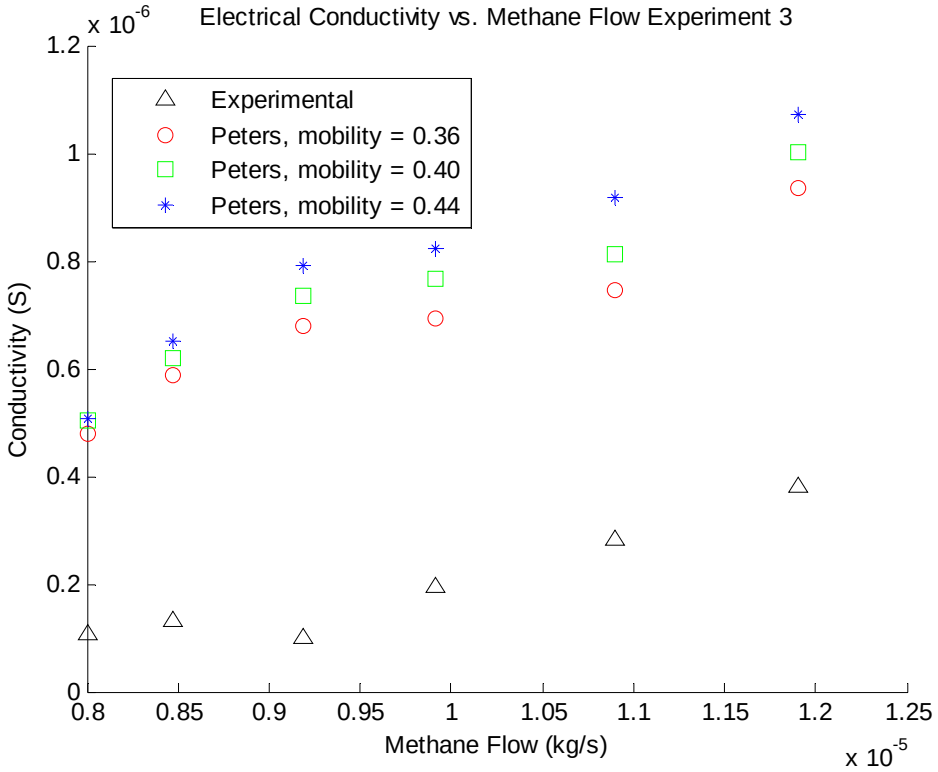

Figure 4-25: Conductivity Using Peters with Methane in Experiment 3

4.1.5 Comparison of All Mechanisms

Among the four reaction mechanisms used in this research, two stand out strongly as being best able to predict methane flame electrical properties. These two reaction mechanism are the Jones, Becker and Heinsohn and the modified GRIMech 3.0. The Peters mechanism shows some potential for use in predicting electrical properties of simulated methane flames, but there is a flaw within the mechanism that results in overprediction of saturation currents. In this particular study, the Pederson and Brown mechanism performed very poorly at predicting flame electrical properties.

Among the three best performing mechanisms, the flame conductivity was consistently overpredicted using mobility values of $0.36\,m^2/Vs$, $0.40\,m^2/Vs$, and $0.44\,m^2/Vs$, but generally, an improvement was apparent as the mobility was decreased, which suggests that a lower mobility value should be used. The performance of the Pederson and Brown mechanism was very poor for electrical properties, and the conductivity values calculated by this mechanism are believed to be unreliable as a result.

The following tables summarize the properties of the V-I curves for simulations run with each mechanism in addition to the experimental findings. Remember that the slope of the V-I curve is equivalent to conductivity.

Experiment Number	Experimental and Modeled V-I Curve Slope to 90% of Peak				
	Experimental	GRIMech 3.0	Jones, Becker and Heinsohn	Pederson and Brown	Peters
1-1	5.215200E-07	6.396600E-07	6.405300E-07	1.003300E-06	1.002300E-06
1-2	6.216900E-07	7.250900E-07	6.966500E-07	3.093700E-07	8.811500E-07
1-3	5.813300E-07	8.455400E-07	7.557800E-07	1.145500E-07	9.016800E-07
1-4	4.779700E-07	8.066600E-07	8.398100E-07	3.308100E-08	8.668700E-07
1-5	3.911800E-07	7.103900E-07	9.861100E-07	2.049000E-08	9.772000E-07
1-6	3.268800E-07	5.593700E-07	1.076800E-06	1.420500E-08	1.055600E-06
2-1	2.301000E-07	1.910400E-06	1.048000E-06	3.997800E-07	4.067400E-06
2-2	1.676900E-07	1.487800E-06	9.006300E-07	1.475900E-07	2.938600E-06
2-3	1.446600E-07	1.253200E-06	8.154000E-07	8.310900E-08	2.369600E-06
2-4	1.287000E-07	1.122400E-06	7.551800E-07	5.134100E-08	2.081900E-06
2-5	1.190400E-07	1.032700E-06	7.115700E-07	3.176200E-08	1.905600E-06
2-6	1.071000E-07	9.317700E-07	6.697900E-07	2.134200E-08	---
3-1	3.824200E-07	6.396600E-07	6.405300E-07	1.003300E-06	1.002300E-06
3-2	2.837100E-07	6.815400E-07	6.655600E-07	2.465600E-07	8.114000E-07
3-3	1.952900E-07	6.457100E-07	6.445300E-07	6.569200E-08	7.666000E-07
3-4	1.010900E-07	4.880900E-07	6.900300E-07	1.539300E-08	7.340300E-07
3-5	1.305300E-07	3.389200E-07	8.269500E-07	7.294900E-09	6.198100E-07
3-6	1.080400E-07	2.273500E-07	9.520100E-07	4.239000E-09	5.022600E-07

Table 4-13: Experimental and Modeled V-I Curve Slopes to 90% of Maximum Current for All Mechanisms Using Methane Combustion with Mobility = 0.40 m^2/Vs

Experiment Number	Experimental and Modeled Voltage at 90% of Peak Current				
	Experimental	GRIMech 3.0	Jones, Becker and Heinsohn	Pederson and Brown	Peters
1-1	59.876	51	40	2	152
1-2	39.878	26	28	1	98
1-3	29.894	14	21	1	60
1-4	29.894	10	16	2	43
1-5	29.894	8	12	2	28
1-6	29.894	8	10	2	20
2-1	-159.96	-23	-38	-11	-43
2-2	-159.95	-18	-35	-5	-35
2-3	-139.95	-14	-32	-3	-28
2-4	-119.95	-11	-30	-2	-22
2-5	-104.95	-9	-28	-2	-18
2-6	-99.949	-8	-27	-2	---
3-1	89.953	51	40	2	152
3-2	69.943	22	23	1	87
3-3	59.947	11	15	1	44
3-4	49.945	8	10	2	26
3-5	39.949	6	6	2	17
3-6	34.946	6	4	2	14

Table 4-14: Experimental and Modeled Voltage at 90% of Maximum Current for All Mechanisms Using Methane Combustion with Mobility = 0.40 m^2/Vs

Experiment Number	Experimental and Modeled Current at 90% of Peak				
	Experimental	GRIMech 3.0	Jones, Becker and Heinsohn	Pederson and Brown	Peters
1-1	3.1549000E-05	3.4428000E-05	2.6630000E-05	2.7251000E-06	1.5499000E-04
1-2	2.4767000E-05	2.0427000E-05	2.0510000E-05	4.5822000E-07	8.8739000E-05
1-3	1.7386000E-05	1.3249000E-05	1.6824000E-05	1.6984000E-07	5.6301000E-05
1-4	1.4281000E-05	9.3561000E-06	1.4376000E-05	9.3304000E-08	3.9306000E-05
1-5	1.1692000E-05	6.9446000E-06	1.2738000E-05	5.7166000E-08	2.9324000E-05
1-6	9.7458000E-06	5.6134000E-06	1.1625000E-05	3.9151000E-08	2.3079000E-05
2-1	-3.5110000E-05	-3.6770000E-05	-2.8158000E-05	-2.6498000E-06	-1.6314000E-04
2-2	-2.7077000E-05	-2.1388000E-05	-2.1530000E-05	-4.7895000E-07	-9.2998000E-05
2-3	-2.0580000E-05	-1.3571000E-05	-1.7429000E-05	-1.7628000E-07	-5.8568000E-05
2-4	-1.5781000E-05	-9.3891000E-06	-1.4966000E-05	-7.5428000E-08	-3.9746000E-05
2-5	-1.2695000E-05	-7.0585000E-06	-1.3029000E-05	-5.0980000E-08	-2.9625000E-05
2-6	-1.0889000E-05	-5.7643000E-06	-1.1881000E-05	-3.7027000E-08	---
3-1	3.5078000E-05	3.4428000E-05	2.6630000E-05	2.7251000E-06	1.5499000E-04
3-2	1.9859000E-05	1.6388000E-05	1.6126000E-05	3.6409000E-07	7.2725000E-05
3-3	1.2334000E-05	8.1736000E-06	1.0537000E-05	9.7015000E-08	3.5466000E-05
3-4	8.1020000E-06	4.7265000E-06	7.5195000E-06	4.3369000E-08	2.0511000E-05
3-5	5.2605000E-06	2.6077000E-06	5.4365000E-06	1.9420000E-08	1.1692000E-05
3-6	3.7985000E-06	1.8452000E-06	4.2400000E-06	1.1549000E-08	8.0254000E-06

Table 4-15: Experimental and Modeled Current at 90% of Maximum for All Mechanisms Using Methane Combustion with Mobility = 0.40 m^2/Vs

A comparision of the some major flame properties for all four mechanisms reveals a few interesting features. One notable feature is that there is a low temperature

and high temperature pairs of mechanisms. This was an unintended effect, but the reason for this effect becomes more apparent when the concentration of species CH is considered. Within flame spectrometry, the presence of CH is often used as an indicator of flame position. Applying this principal to the simulated flame objects reveals that the colder mechanisms, Jones, Becker and Hiensohn and Pederson and Brown, have flame positions much closer to the burner than the warmer mechanisms, GRIMech 3.0 and Peters. Both in reality and in the simulation, the water cooled burner acted as a heat sink, which means a flame that is positioned closer to the burner will loose more heat to the burner than one that burns further away. The next three figures show peak flame temperatures for each mechanism, and are immediately followed by plots of the distance from the burner of the maximum CH concentration for each test case.

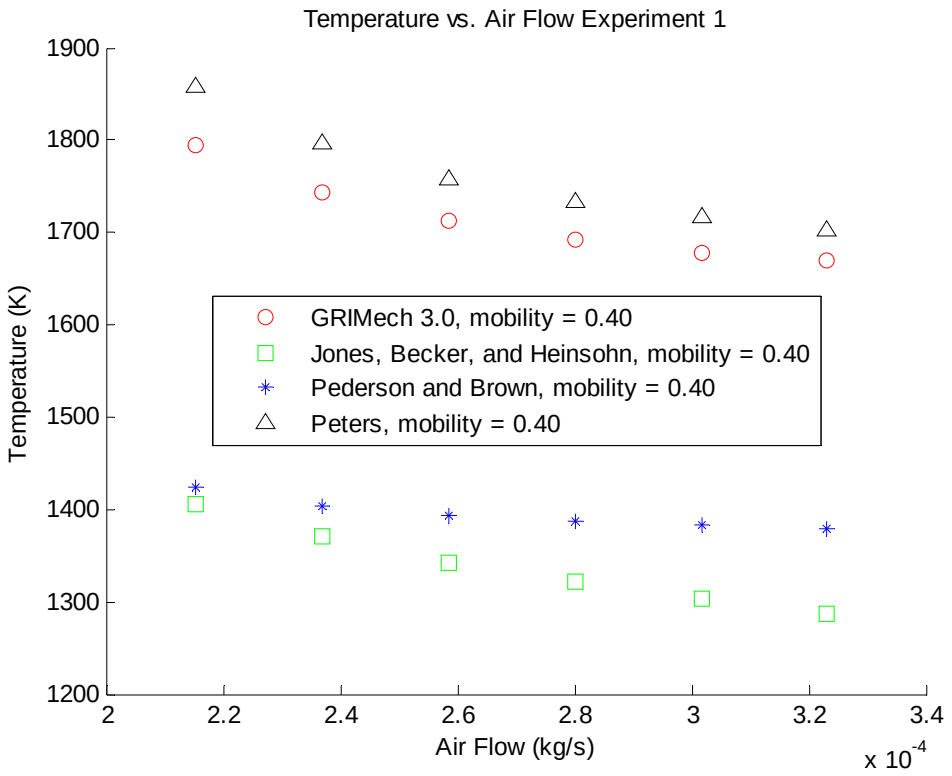

Figure 4-26: Temperature vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 1 with Mobility = 0.40 m^2/Vs

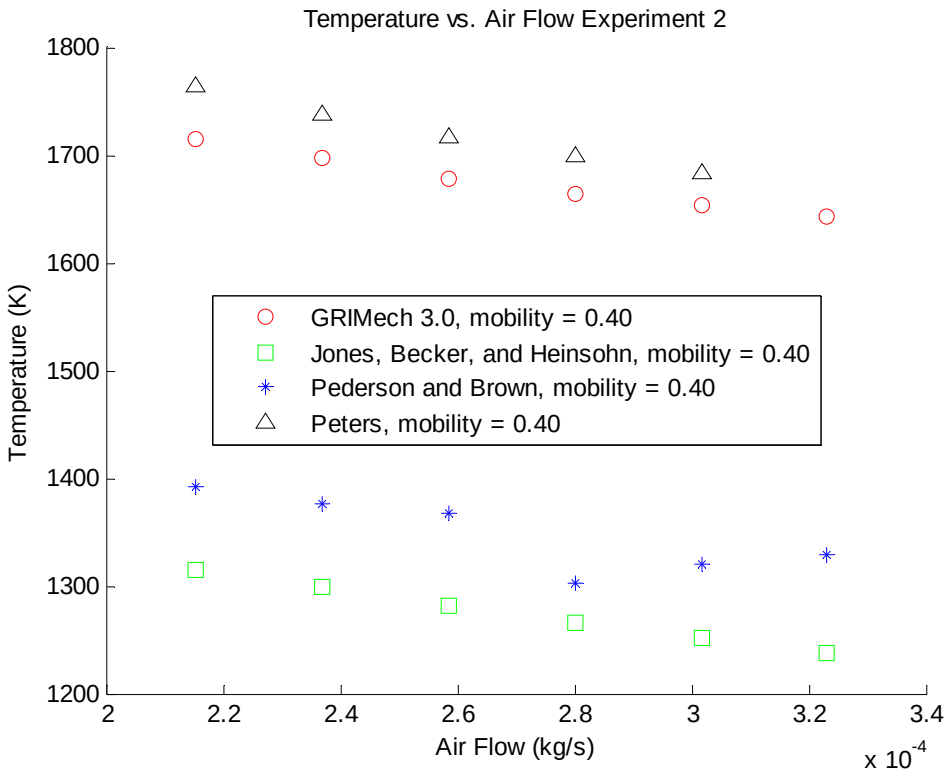

Figure 4-27: Temperature vs. Methane Flow for Methane Combustion Using All Mechanisms in Experiment 2 with Mobility = 0.40 m^2/Vs

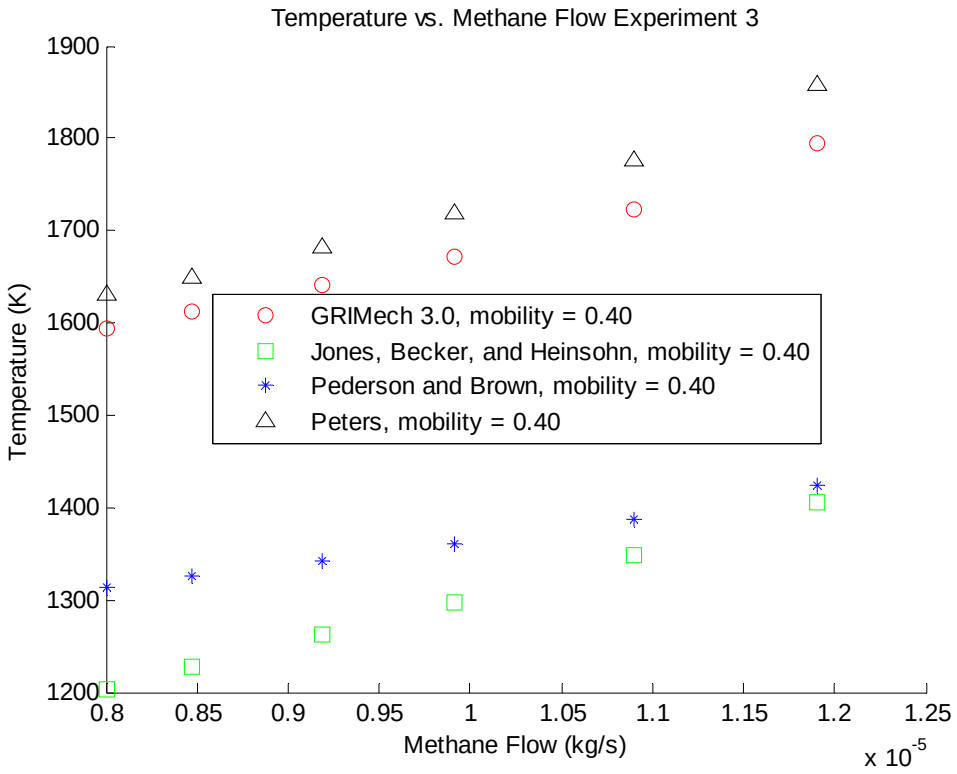

Figure 4-28: Temperature vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 3 with Mobility = 0.40 m^2/Vs

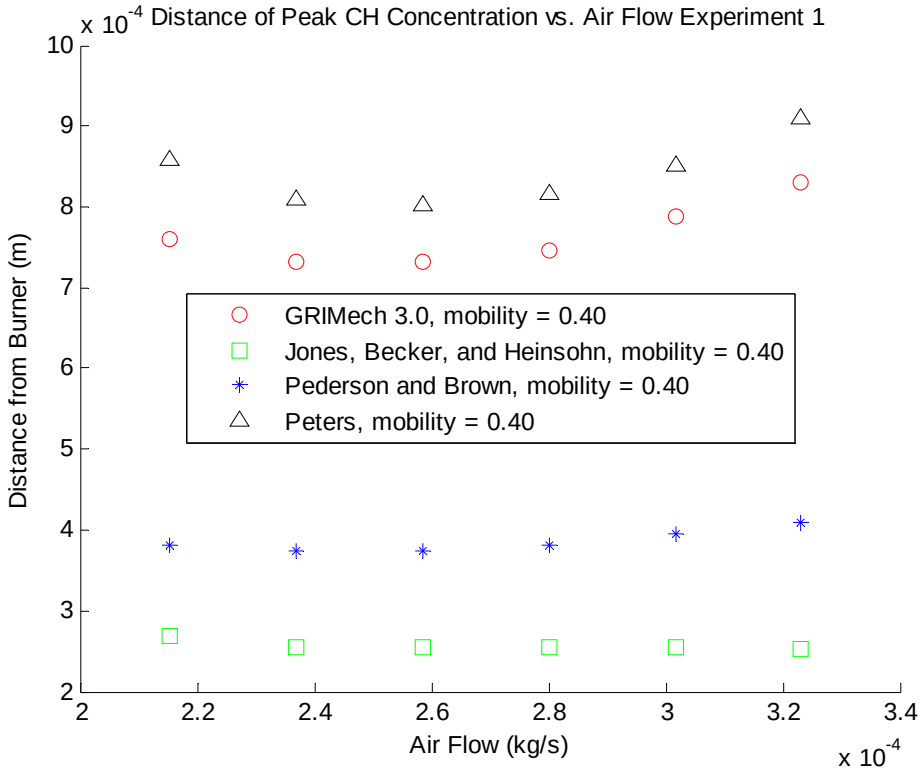

Figure 4-29: Position of CH Peak Consentration vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 1with Mobility = 0.40 m^2/Vs

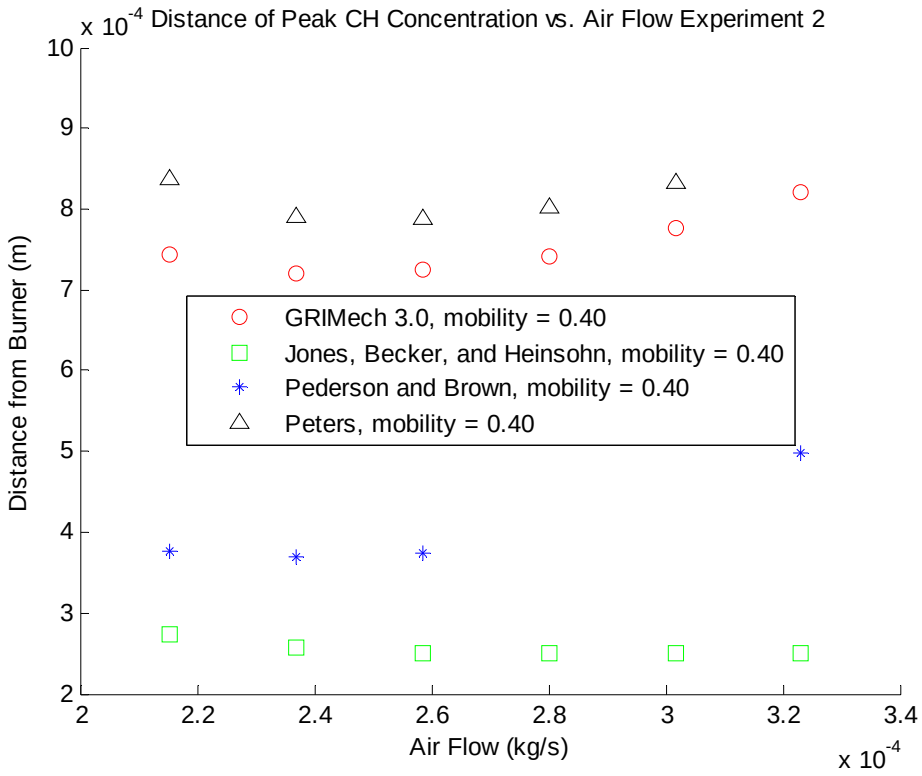

Figure 4-30: Position of CH Peak Consentration vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 2 with Mobility = 0.40 m^2/Vs

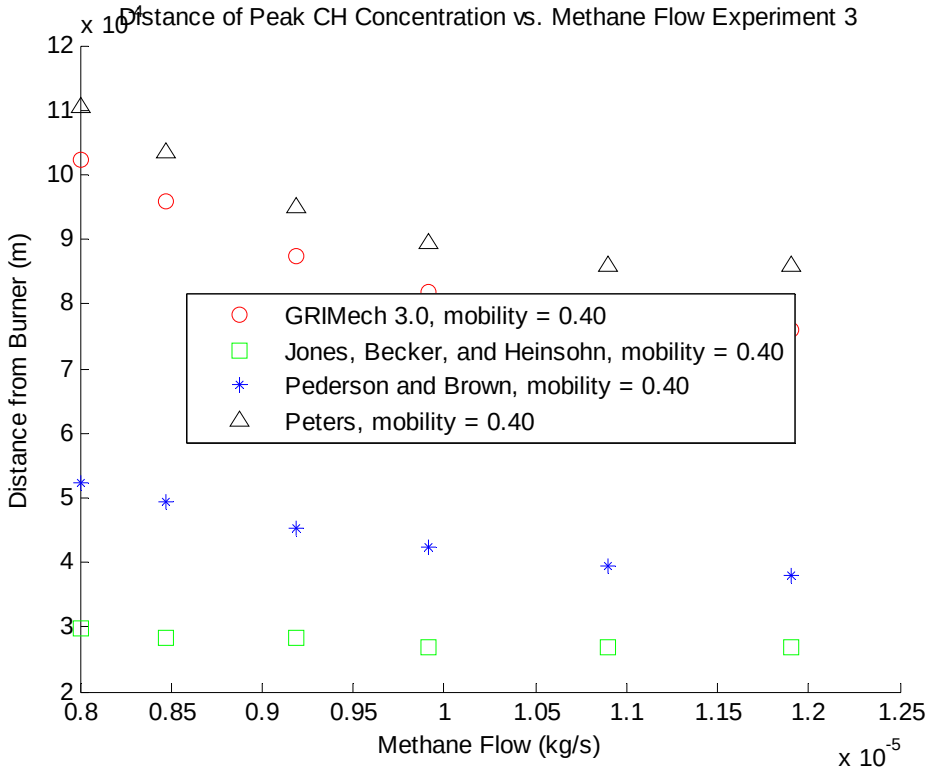

Figure 4-31: Position of CH Peak Consentration vs. Methane Flow for Methane Combustion Using All Mechanisms in Experiment 3 with Mobility = 0.40 m^2/Vs

Another interesting point concerning CH is the fact that it is a precursor molecule for all of the positive species HCO^+ and $H3O^+$. This could explain the problem with overestimation of saturation current by the Peters mechanism because the concentration of CH generated by this mechanism was significantly higher than in all other mechanisms. This could result in overproduction of charged species and, as a result, overprediction of current. An example of the CH concentrations is shown in the following figure.

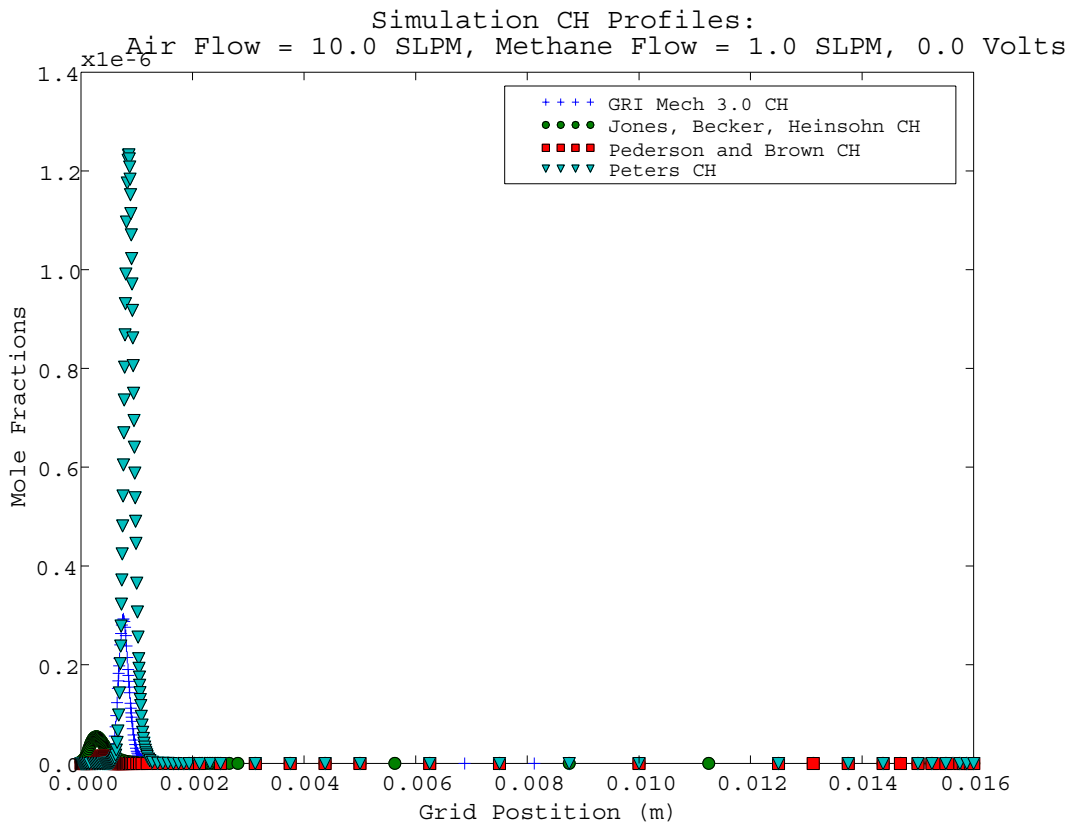

Figure 4-32: Example of Typical CH Profiles for Each Mechanism During Methane Combustion

The figures below show a comparison of the experimental and simulated saturation currents for each mechanism. The Jones, Becker and Heinsohn and GRIMech 3.0 both show good agreement with the experimental data.

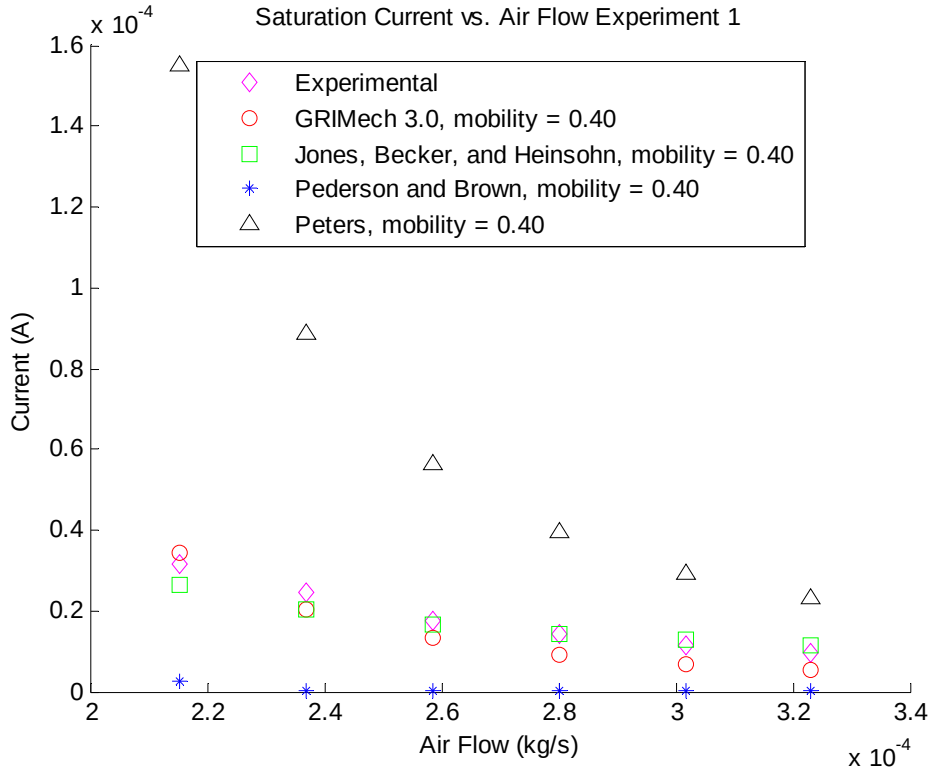

Figure 4-33: Saturation Currents vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 1 with Mobility = 0.40 m^2/Vs

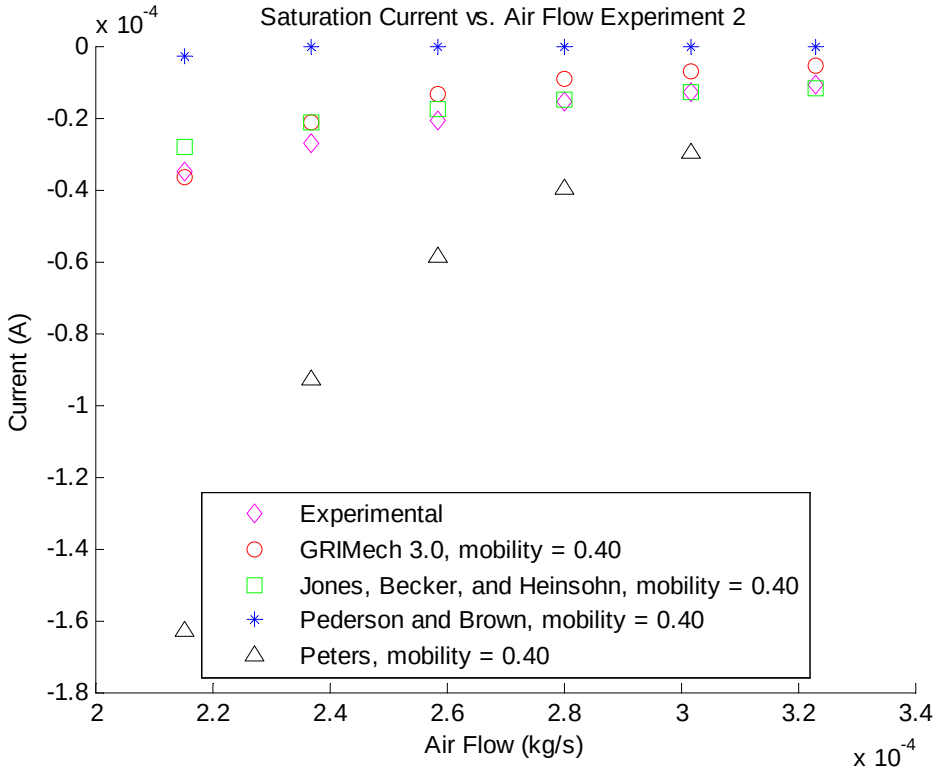

Figure 4-34: Saturation Currents vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 2 with Mobility = 0.40 m^2/Vs

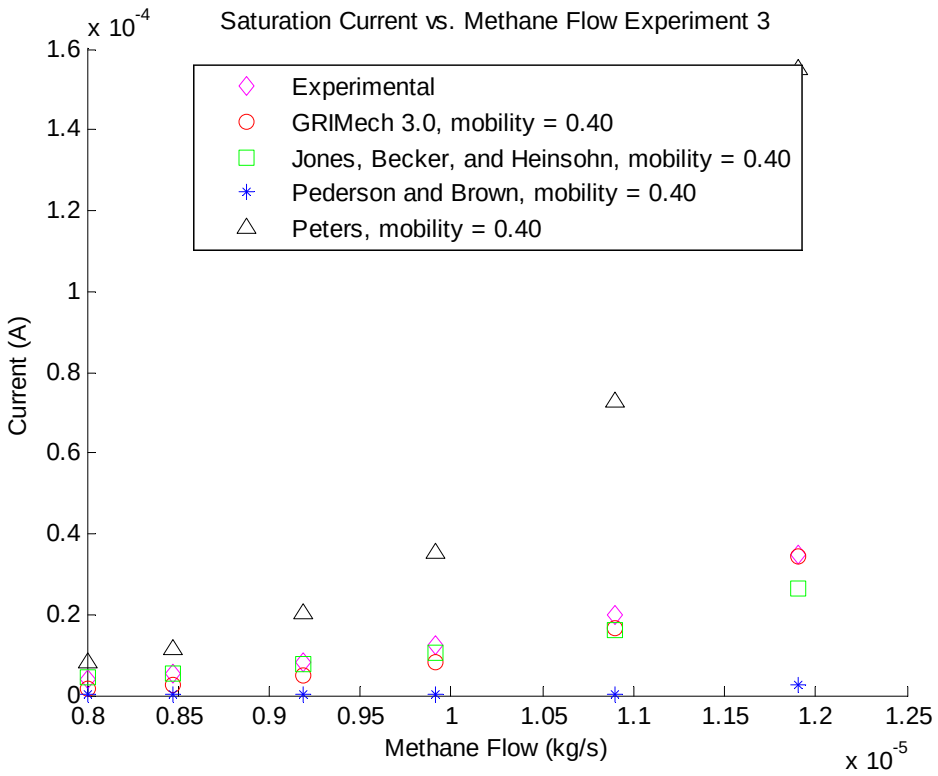

Figure 4-35: Saturation Currents vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 3 with Mobility = 0.40 m^2/Vs

For all mechanisms except Pederson and Brown, the saturation current is almost always significantly to high. This fact is shown in the following plots.

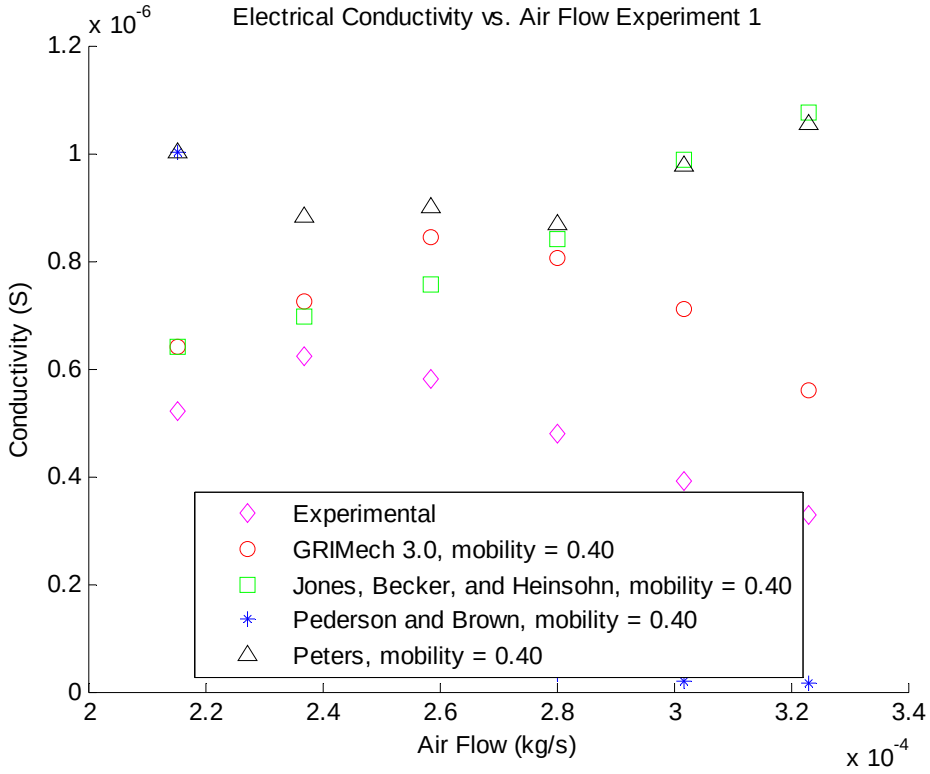

Figure 4-36: Conductivity vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 1 with Mobility = 0.40 m^2/Vs

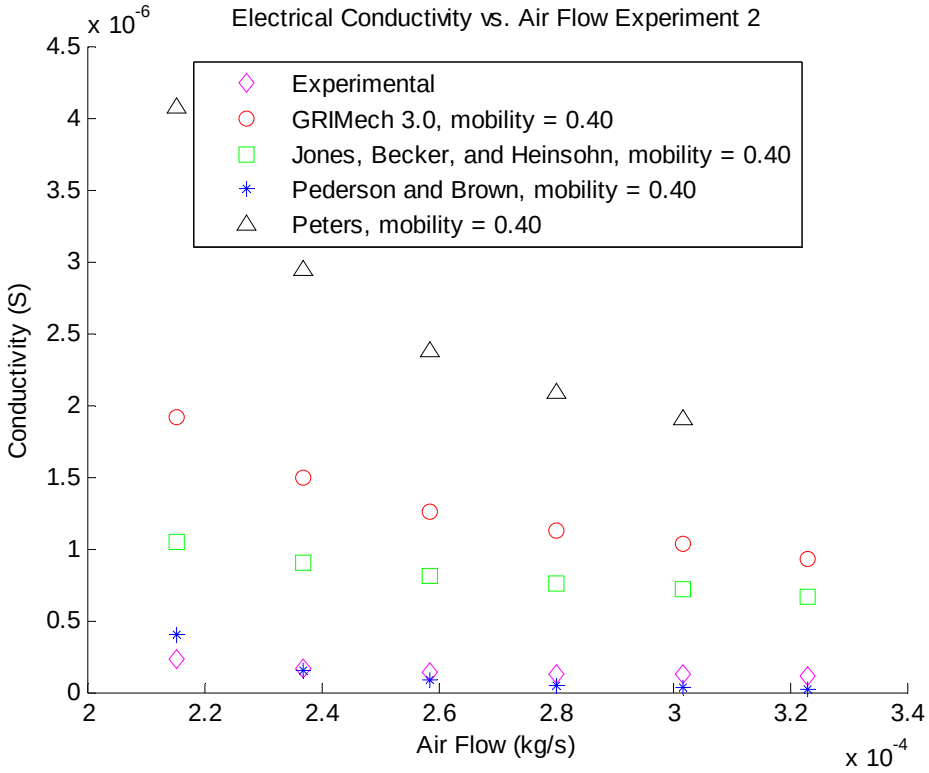

Figure 4-37: Conductivity vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 2 with Mobility = 0.40 m^2/Vs

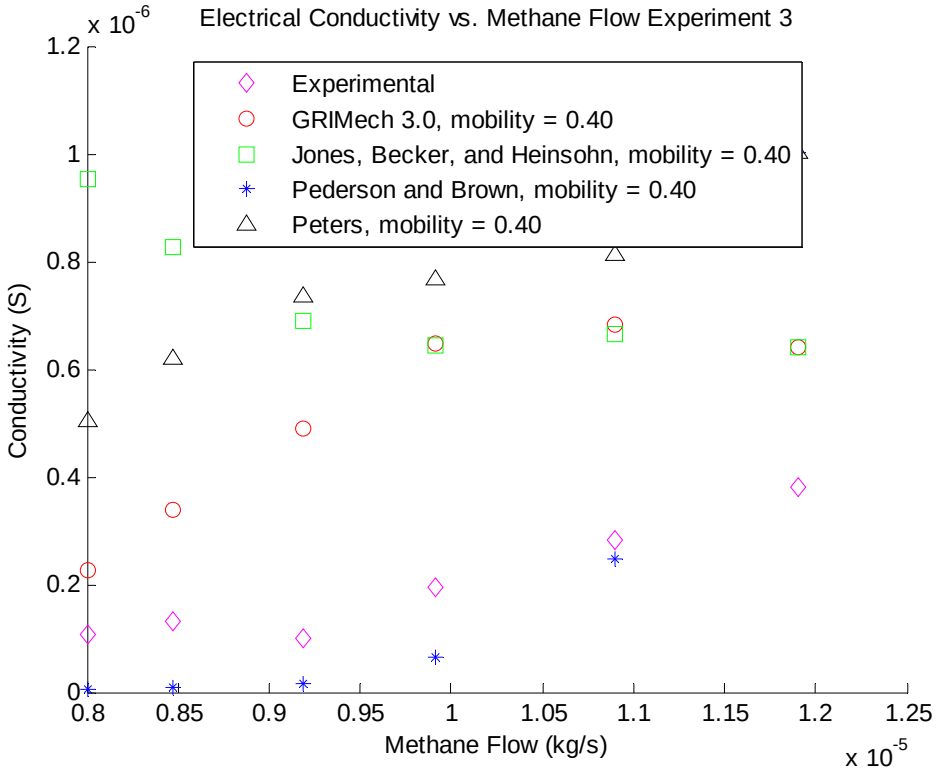

Figure 4-38: Conductivity vs. Air Flow for Methane Combustion Using All Mechanisms in Experiment 3 with Mobility = 0.40 m^2/Vs

4.2 Synthesis Gas Models

For the synthesis gas models, only one reaction mechanism was used. This mechanism was the modified version of the GRIMech 3.0 which is discussed in Chapter Two, and the listing of the Cantera 'cti' file containing this mechanism is given in Appendix B. The choice to use only this reaction mechanism was made because of extreme difficulty achieving convergence with the other reaction mechanisms mentioned in this paper using synthesis gas. One likely explanation for this problem is the fact that the mechanisms used in this research had been designed for use in natural gas combustion, but the only hydrocarbon within the combustion gas mix for the synthesis gas experiments was the varying, but small amounts of methane added in order to increase the generation of charged molecules formed during combustion. A more ideal reaction mechanism for synthesis gas would place a higher focus on reactions involving hydrogen gas and carbon monoxide, the main constituents of synthesis gas.

As with the methane models, the synthesis gas models were run using three different mobilities: 0.36, 0.40, and 0.44 m^2/Vs. With all tested mobilities, and with all mixtures of methane added, the simulated saturation current was underpredicted. Also, the conductivity found in the model was extremely high. The low saturation current and high conductivity resulted in saturation at extremely low voltages in the simulated flame. The major V-I curve properties can be seen in the tables below.

Run Number	Pederson and Brown Synthesis Gas Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.36 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1	3.8115E-08	3.7116E-08	8.2008E-08	-5.6992E-08	0.75	-139.43
2	5.5204E-08	5.2624E-08	2.3173E-07	-8.3270E-08	2.00	-117.85
3	1.2312E-07	1.1878E-07	4.4304E-07	-1.1496E-07	2.00	-127.85
4	1.7234E-07	1.6657E-07	5.7791E-07	-1.7109E-07	2.00	-128.19
5	1.7204E-07	1.6289E-07	7.9091E-07	-1.2809E-07	3.00	-96.866

Table 4-16: V-I Curve Characteristics to 90% of Maximum Current Using GRIMech 3.0 with Electron Mobility of 0.36 m^2/Vs for Synthesis Gas Combustion

Run Number	Pederson and Brown Synthesis Gas Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.40 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1	3.6205E-08	3.5206E-08	8.2536E-08	-5.6464E-08	0.75	-139.43
2	5.2548E-08	4.9968E-08	2.3243E-07	-8.2570E-08	2.00	-117.85
3	1.1821E-07	1.1387E-07	4.4534E-07	-1.1266E-07	2.00	-127.85
4	1.6684E-07	1.6107E-07	5.8218E-07	-1.6682E-07	2.00	-128.19
5	2.2335E-07	2.1420E-07	7.4068E-07	-1.7832E-07	2.00	-97.866

Table 4-17: V-I Curve Characteristics to 90% of Maximum Current Using GRIMech 3.0 with Electron Mobility of 0.40 m^2/Vs for Synthesis Gas Combustion

Run Number	Pederson and Brown Synthesis Gas Combustion V-I Curve Characteristics to 90% of Peak Current with Mobility = 0.44 m^2/Vs					
	Slope		Current		Voltage	
	Modeled	Difference	Modeled	Difference	Modeled	Difference
1	3.4469E-08	3.3470E-08	8.2981E-08	-5.6019E-08	0.75	-139.43
2	5.0170E-08	4.7590E-08	2.3296E-07	-8.2040E-08	2.00	-117.85
3	1.1381E-07	1.0947E-07	4.4722E-07	-1.1078E-07	2.00	-127.85
4	1.6149E-07	1.5572E-07	5.8570E-07	-1.6330E-07	2.00	-128.19
5	2.1722E-07	2.0807E-07	7.4645E-07	-1.7255E-07	2.00	-97.866

Table 4-18: V-I Curve Characteristics to 90% of Maximum Current Using GRIMech 3.0 with Electron Mobility of 0.44 m^2/Vs for Synthesis Gas Combustion

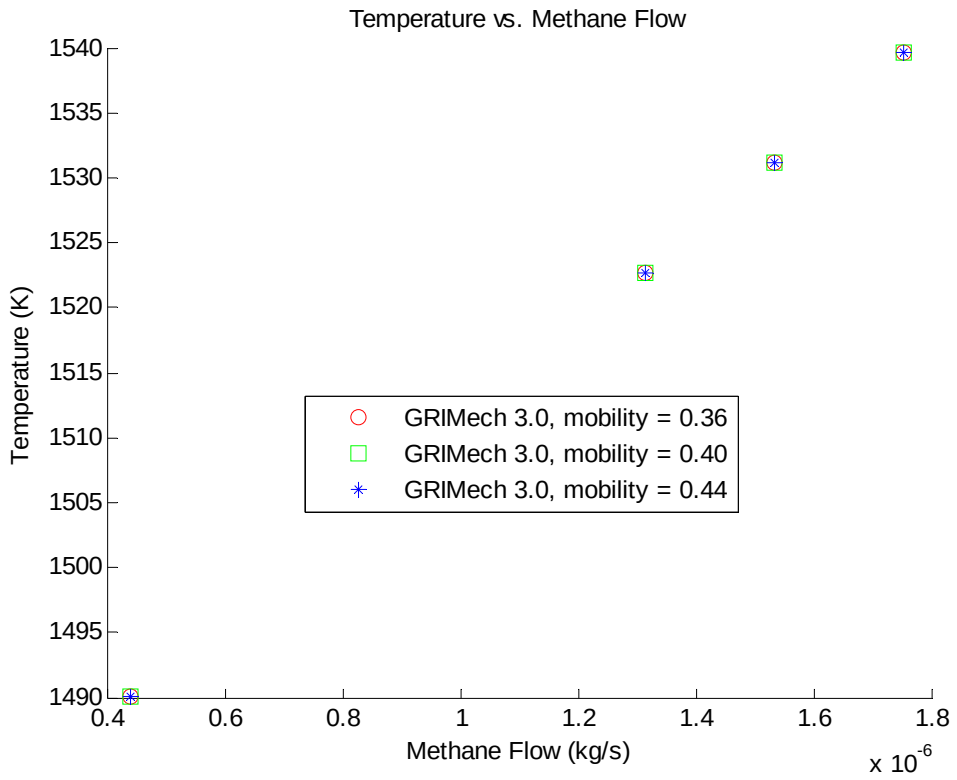

Figure 4-39: Temperature vs. Methane Flow Using GRIMech3.0 for Methane-Doped Synthesis Gas Combustion

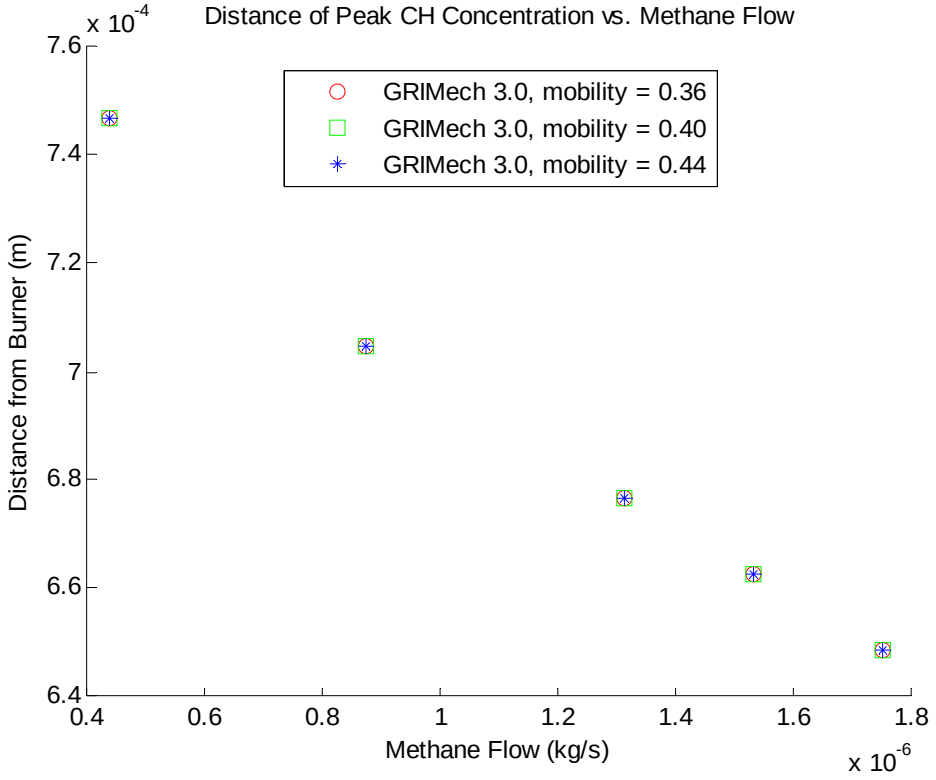

Figure 4-40: Position of Peak CH Concentration vs. Methane Flow Using GRIMech3.0 for Methane-Doped Synthesis Gas Combustion

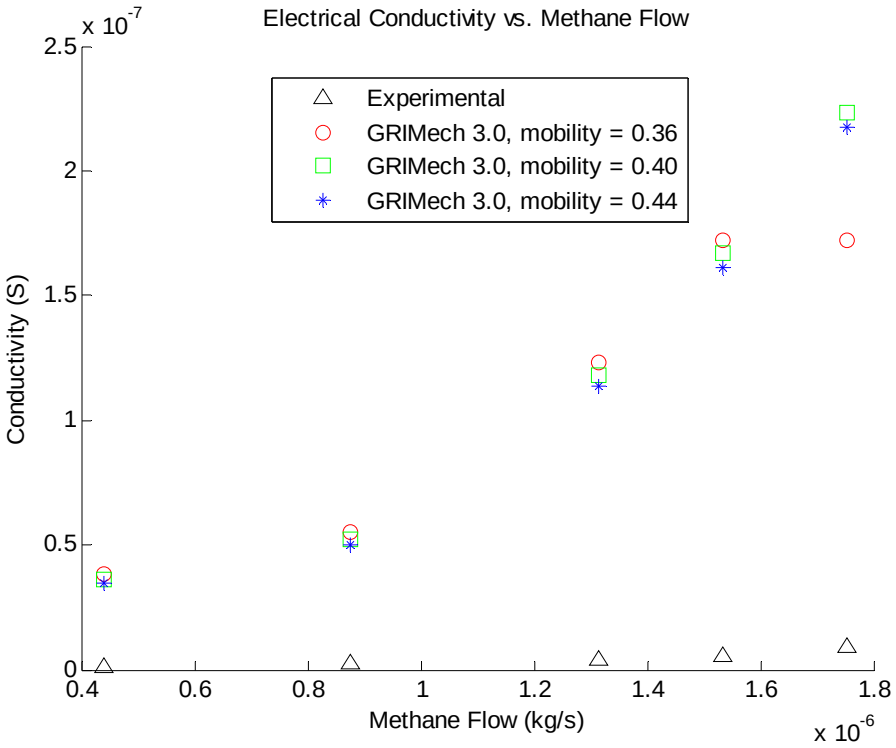

Figure 4-41: Conductivity vs. Methane Flow Using GRIMech3.0 for Methane-Doped Synthesis Gas Combustion

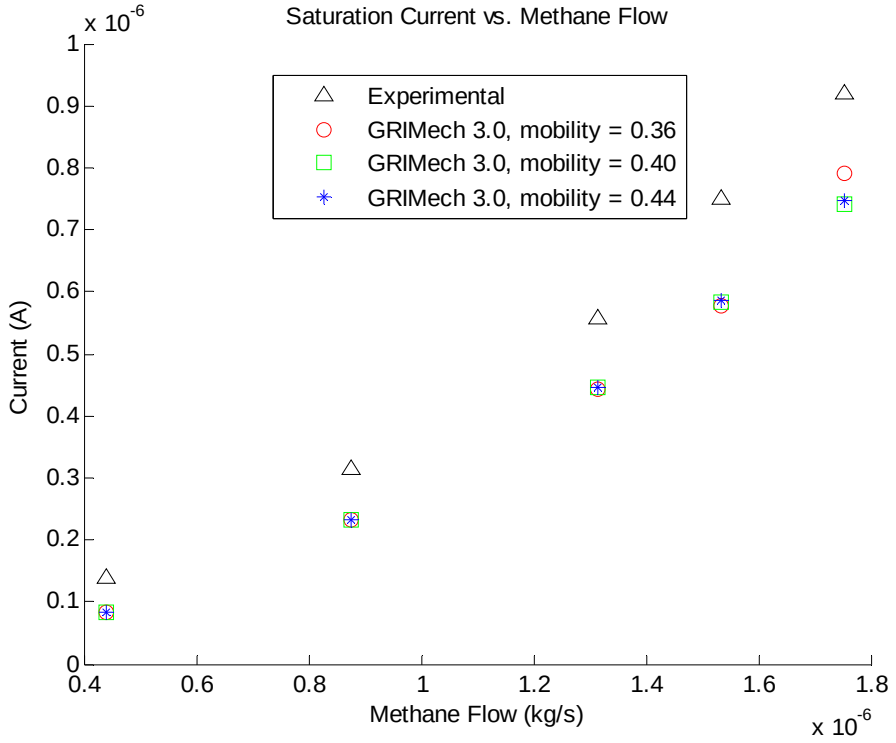

Figure 4-42: Saturation Current vs. Methane Flow Using GRIMech3.0 for Methane-Doped Synthesis Gas Combustion

In the preceeding figures, a few tendancies can be seen. As the methane concentration increases, the temperature of the simulated flame increases and the distance between the burner and the flame decreases. As predicted experimentally, the conductivity increases with methane concentration, but the conductivity values are overestimated in all cases. Also, the saturation currents found are not shown to be extremely far off the experimental values, but are consistently overpredicted.

Because of the difficulties with convergence for three of the mechanisms in this research and questionable results from the fourth, it is believed that a completely different mechanism will be needed to accurately predict synthesis gas flame electrical properties.

Chapter Five: Conclusion

This work modifed and extended a one-dimensional chemical kinetics simulation code to incorporate the effects of charged species and applied electric fields. This code was then used to compare existing experimental data with simulation results calculated with the use of four different published reaction mechanisms.

This code was based upon the Cantera chemical kinetics simulation code and was extended to incorporate the effects of applied electric fields. By incorportating Gauss' Law, this extension included the ability to specify an applied electric potential and allow for movement of charged species within the simulated flame. This movement in addition to diffusion now includes drift caused by the electric field. Analysis of simulation results using charged species demonstrated rising current with rising voltage until reaching a maximum charge carrier velocity. This saturation current effect has been demonstrated experimentally.

This code requires chemical reaction mechanisms as input. A reaction mechanism is simply sets of all possible reactions, elements, chemical species, and respective properties. Four published reaction mechanisms were tested.

Two of these mechanisms, modified *GRIMech 3.0* and *Jones, Becker and Hiensohn*, found saturation values close to the experimental values. Although both mechanisms exhibit different temperatures and flame position, the charged species reactions interacted with neutral species reactions and transport to produce similar results largely due to the position and quantity of CH, a major precursor to charged species generation. The *Peters* mechanism using the same charged reactions and species, consistently overpredicted saturation currents because of overprediction of CH. The

Pederson and Brown mechanism consistently underpredicted currents and proved to be unstable.

All simulated stable mechanisms seemed to overpredict the flame conductivity when compared to the available measured data. This result indicates that the value of 0.4 m^2/Vs for the electron mobility too high, which is not in agreement with the paper by Goodings et al., the source used for the electron mobility value. Decreasing electron mobility to $0.36\, m^2/Vs$ improved the calculated conductivity, but even that value overpredicted conductivity.

Chapter Six: Future Work

From the results seen so far in this research, some ideas for future work are apparent. One point that stands out is the need to attempt simulations with lower values for electron mobility. Given the magnitude of the error in electrical conductivity, the use of a value as low as $0.04\,{}^{m^2}\!/_{Vs}$, which is one tenth of the base value used in this research, may not be unreasonable. Secondly, further reaction mechanism refinement for charged particle chemistry should prove extremely useful for improving the accuracy of the simulation predictions for saturation currents. This refinement should include the addition of other less numerous charged species in addition to electrons, HCO+ and H3O+, especially negative charge carriers. In these models, the electrons were used as the only negative carriers, but their small size and weight allows them to travel significantly faster than the molecule sized positive charge carriers.

In addition, additional verification of the original experimental setup my be very useful. Some significant parameters such as space charge, high temperature effects on metal electrodes, and oxidation of the electrode may not have been taken fully into account.

Appendix A: Driver Codes

A-1: Methane Combustion Driver Code

Description:

Although the modified version of Cantera is capable of running simulations in the same manner as the standard version of Cantera, an additional python program has been written in order to simplify the task of running multiple similar simulations that differ only by applied voltage. The `EFSim_NG_Current` program has been written in order to allow for easy reuse of the flame objects while running these similar simulations. By allowing for reuse of flame objects, the computation time can be greatly reduced because the system only needs to acquire a full solution to one case in a set of similar simulations, and the solutions to the remaining case may be found by using the fully solved case as a starting point for finding the solutions to the remaining cases.

Reuse of flame objects is computationally advantageous when the major properties of the flame are not expected to change significantly between simulated cases. This advantage can be used for these simulations because the only species which are expected to change greatly as a result of changing the applied voltages are the charged species, which make up an extremely small fraction of the mass within the flame. The assumption that neutral species, flame position, and temperature profile do not change significantly by changing the applied voltage is based on the use of reasonably small voltages. If the applied voltage through the flame is increased to the order of kilovolts, then the effects of ionic wind would need to be taken into account which would void this assumption. For the purposes of the CCADS project, voltages high enough to generate a

significant ionic wind are not necessary. As a result, ionic wind effects have not been included in this model.

Finally, although the code does not enforce the use of small voltage steps when changing voltage, the use of small steps in voltage level is recommended and used in this research in order to further save computation time. For example, all simulations demonstrated in this document were produced by starting at zero applied voltage. Multiple small voltage increases were made between zero and one volt, and only one volt increments were used thereafter.

Code Overview:

All of the code used to streamline methane combustion simulation is included in a set of two classes. The `EFSim_NG_Current` class holds all of the functions needed to setup a simulation given the equivalence ratio and mass flow rates of the premixed gas used for combustion. The `EFSim_NG_Current_mass` class extends the `EFSim_NG_Current` class to allow for simulations with known flow rates of air and fuel in the premixed gas. This extension adds only an additional constructor for use in setting up a simulation.

The following table gives a listing of the constructor inputs for each of the two classes.

Class	Input Name	Purpose	Units
EFSim_NG_Current	phi	Equivalence ratio of the desired gas mixture	---
	m_dot_total	Mass flux of air and fuel combined.	kg/s
	area	Burner area of simulated flame object	m^2
	rxmech	Filename of the Cantera "cti" file containing the desired reaction mechanism file.	---
	rxmix	Name of the mix type within the reaction mechanism file to use for the simulation.	---
	use_efield	Boolean flag used to denote if an applied electric field will be used in the desired simulation.	---
	applied_voltage	Initial applied voltage for the simulation object.	V
	current_file	Name of file to use for current calculation performed while running the flame object.	---
EFSim_NG_Current_mass	m_dot_air	Mass flux of air in the methane and air mix.	kg/s
	m_dot_fuel	Mass flux of fuel in the methane and air mix.	kg/s
	area	Burner area of simulated flame object	m^2
	rxmech	Filename of the Cantera "cti" file containing the desired reaction mechanism file.	---
	rxmix	Name of the mix type within the reaction mechanism file to use for the simulation.	---
	use_efield	Boolean flag used to denote if an applied electric field will be used in the desired simulation.	---
	applied_voltage	Initial applied voltage for the simulation object.	V
	current_file	Name of file to use for current calculation performed while running the flame object.	---

Table A-1: Constructor Inputs for Methane Combustion Modeling Code.

Within the EFSim_NG_Current class a set of "constants" is included. Howerver, these values may be easily modified by the user, and the responsibility of leaving these values constant is given to the program's user. These constants are listed in the table below.

Constant Name	Purpose	Value
N2_PERCENT	Percent by mass of N_2 in air	0.76708
O2_PERCENT	Percent by mass of O_2 in air	0.23292
SAFR	Stoichiometric Air Fuel Ratio by mass	17.0194
N2_MW	Molecular weight of N_2	28.0134
O2_MW	Molecular weight of O_2	31.9998
FUEL_MW	Molecular weight of CH_4	16.13782

Table A-2: Constants Included in the `EFSim_NG_Current` Class

The remainder of the functions used for setting up and running simulations are include directly within the `EFSim_NG_Current` class, but are accessible and useable by the `EFSim_NG_Current_mass` class. These functions are described in the table below.

Function	Purpose	Input	Description	Units
__init__	Class constructor	*see Table A-1		
__del__	Class destructor used to close all open file before disposing of simulation object	---	---	---
initialize	Used to initialize a BurnerFlame or BurnerFlame_efield object for use in simulation	pressure	Gas pressure to use in the simulation	Pascal
		temperature	Burner temperature to use in the simulation	K
		initial_grid	List of the initial set of grid points to use in the simulation.	m
		refine_grid	Indicates if a simulation should attempt to add grid points as needed. 1 indicates that points may be added, and 0 indicates that points should not be added	---
		log_level	Indicates the level of diagnostic output to produce during a simulation. 1 is the minimum, and 5 is the maximum	---
		tol_ss	Steady state error tolerance.	---
		tol_ts	Transient state error tolerance	---
		max_jacobian	Used to limit the reuse of jacobian matrices during calculations	---
		refine_ratio	Ratio used to determine a need for solution refinement	---
		refine_slope	Slope used to	---

			determine a need for solution refinement	
		refine_curve	Curve used to determine a need for solution refinement	---
		time_step	Length of time steps to use while finding steady state solutions	s
		time_refine	List of number of timesteps to take at a time while finding a solution	---
		electron_mobility	Electron mobility to be used in a simulation	m^2/Vs
		init_param	Normalized starting flame position used to aid in flame calculation by allowing an improved initial guess.	---
setElectronModel	Function used to set the electron mobility within the flame object	---	---	---
setFlame	Function used to allow for the use of an externally created BurnerFlame or BurnerFlame_efield object for a simulation.	new_flame	The BurnerFlame or BurnerFlame_efield object to use in a simulation.	---
createFlame	Function intended for internal use in creating a BurnerFlame or BurnerFlame_efield object	---	---	---
setOptionalProperties	Function intended for internal use in setting optional	---	---	---

	properties given as inputs to the initialize function			
setElectricField	Function intended for internal use in initially setting up an electric field for the flame object	---	---	---
initFlame	Used to call Cantera's initialize function on the flame object and flag the flame object as initialized	---	---	---
presolve	This function solves the flame object with the energy equations turned off in order to derive a better initial solution before running a true simulation	---	---	---
solve	This function solves the flame object with the energy equation turned on and saves the resulting output files.	xml_dir	Name of directory in which to store results in XML file format	---
		csv_dir	Name of directory in which to store results in CSV file format	---
change_voltage	Function used to change the voltage between runs of similar simulations	new_volt	Value for the new applied voltage	V
__str__	Function used to display class constants and variables	---	---	---

Table A-3: Class Methods for `EFSim_NG_Current Class`

As a result of running simulations with these programs, three different types of output files are generated. The first type is a solution in Cantera's 'ctml' file format. The second type is also a solution, but this solution is in comma separated value ('csv')

format. The third potential output file is a file listing the currents for each simulation run. All three of these files formats are documented in Appendix D. The names for the first two files are generated using the `nameParse` class, which is specified in Appendix C.

Code Listing:

```
from Cantera import *
from Cantera.OneD import *
from Numeric import *
import nameParse as np

class EFSim_NG_Current:
    """
    This class holds data and methods for a single flame simulation.
The constructor for this class uses Total
    Mass Flux and Equivalence Ratio.  To use Fuel and Air Mass Flux
seperately, use the EFSim_NG_mass class instead.
    """
    #Constant
    N2_PERCENT = 0.76708 #percent by mass N2 in air
    O2_PERCENT = 0.23292 #percent by mass O2 in air
    SAFR = 17.0194 #Stoicheometric Air/Fuel Ratio
    O2_MW = 31.9998 #Oxygen molecular weight
    N2_MW = 28.0134 #Nitrogen molecular weight
    FUEL_MW = 16.13782 #Fuel Molecular Weight

    #Variable
    applied_voltage = None
    area = None
    flame = None
    flow_rate = None
    gas = None
    initial_grid = None
    log_level = None
    max_jacobian = None
    m_dot_air = None
    m_dot_fuel = None
    m_dot_total = None
    mass_fractions = None
    phi = None
    pressure = None
    refine_curve = None
    refine_grid = None
    refine_ratio = None
    refine_slope = None
    rxmech = None
    rxmix = None
    time_step = None
    time_refine = None
    tol_ss = None
    tol_ts = None
    temperature = None
    use_efield = None
```

```
    current_file = None
    modelFactor = None
    name_gen = None
    init_param = None

    def __init__(self, phi, m_dot_total, area, rxmech, rxmix,
use_efield = False, applied_voltage = None, current_file = None):
        """
        Constructor used to generate EFSim object.
        Based on known total mass flux and equivalence ratio.

        phi = Equivalence Ratio
        m_dot_total = mass flux of air + mass flux of CH4 in kg/s
        area = burner area in m^2
        rxmech = reaction mechanism file
        rxmix = mix type from mechanism file
        use_efield = True -- use Cantera.BurnerFlame_efield
                     False -- use Cantera.BurnerFlame
        applied_voltage = applied voltage (Volts)
        current_file = name of file in which to store current values
        """
        #setup class data
        self.phi = phi
        self.m_dot_total = m_dot_total
        self.area = area
        self.use_efield = use_efield
        self.applied_voltage = applied_voltage
        self.flow_rate = m_dot_total / area
        self.rxmech = rxmech
        self.rxmix = rxmix

        #determine mix
        afr = self.SAFR/self.phi
        fuel_per = 1.0
        air_per = afr
        sum_val = fuel_per + air_per
        air_per = air_per/sum_val
        fuel_per = fuel_per/sum_val

        fuel_mass = fuel_per * self.m_dot_total
        n2_mass = air_per * self.N2_PERCENT * self.m_dot_total
        o2_mass = air_per * self.O2_PERCENT * self.m_dot_total

        self.m_dot_air = (n2_mass + o2_mass)
        self.m_dot_fuel = (fuel_mass)
        self.mass_fractions = "N2:" + str(n2_mass) + ",O2:" +
str(o2_mass) + ",CH4:" + str(fuel_mass)

        #test for erroneous use of efield
        if (applied_voltage == None and use_efield == True):
            print "Warning, Applied Voltage declared, but
BurnerFlame_efield is not selected for use."

        if current_file != None:
            self.current_file = open(current_file, 'w')
```

```
self.current_file.write('ER,total_flux,air_flux,fuel_flux,voltage,curre
nt\n')
        self.name_gen = np.nameParse()

    def __del__(self):
        """
        Destructor for EFSim_NG_Current.
        """
        if self.current_file != None:
            self.current_file.close()

    def initialize (self, pressure, temperature, initial_grid,
                    refine_grid = 1, log_level = 5,
                    tol_ss = None, tol_ts = None,
                    max_jacobian = None,
                    refine_ratio = None, refine_slope = None,
                    refine_curve = None,
                    time_step = None, time_refine = None,
                    electron_mobility = 0.4, init_param = None):
        """
        This function is used to initialize a BurnerFlame or
BurnerFlame_efield.
        Inputs as provided to Cantera are required.

        pressure = gas pressure
        temperature = burner temperature in Kelvin
        initial grid = list of points to start evaluation
        refine_grid = 1 to refine the grid when solvin, 0 otherwise
        log_level = level of detail in cantera output, 1 is the
minimum, 5 is the maximum
        tol_ss = steady state tolerance (list of 2 values)
        tol_ts = transient state tolerance (list of 2 values)
        max_jacobian = maximum jacobian ages.  (tuple of 2 values)
        refine_ratio = ratio to adapt refine criteria
        refine_slope = slope to adapt refine criteria
        refine_curve = curve to adapt refine criteria
        time_step = time in seconds for a single timestep
        time_refine = list of number of timesteps to use when refining
        electron_mobility = constant value for the electron mobility
                            -- default value = 0.4
        init_param = normalized starting flame position

        pressure, temperature, and initial_grid must be set.
        All other attributes will be set to default values if not
specified.
        In most cases, the default value is the Cantera default, but
for electron
        mobility, the default value is 0.4.
        """

        #store attributes
        self.pressure = pressure
        self.temperature = temperature
        self.initial_grid = initial_grid
        self.refine_grid = refine_grid
        self.log_level = log_level
```

```
        self.tol_ss = tol_ss
        self.tol_ts = tol_ts
        self.max_jacobian = max_jacobian
        self.refine_ratio = refine_ratio
        self.refine_slope = refine_slope
        self.refine_curve = refine_curve
        self.time_step = time_step
        self.time_refine = time_refine
        self.modelFactor = electron_mobility
        self.init_param = init_param
        #perform setup
        self.createFlame()
        self.setOptionalProperties()
        self.setElectronModel()
        self.setElectricField()
        self.initFlame()

    def setElectronModel(self):
        """
        This function is used to set the electron model of all charged
species.
        High probability for change.
        """
        if self.use_efield == True:
            electronModel = self.flame.flame.emodel_CONST
            useMassGradient = 0
            self.flame.flame.setElectronTransModel(electronModel,
useMassGradient, self.modelFactor)
            for index in range(0,self.flame.gas.nSpecies()):
                if self.flame.gas.nAtoms(index,'E') < 0:
                    self.flame.burner.setAbsorbtionFactor(index,1.0)

self.flame.burner.setSpeciesBcType(index,self.flame.burner.BC_THERMAL)
### Perhaps BC_Value

self.flame.outlet.setSpeciesBcType(index,self.flame.burner.BC_VALUE)
                elif self.flame.gas.nAtoms(index,'E') > 0:
                    self.flame.burner.setSpeciesBcType(index,
self.flame.burner.BC_VALUE)
                    self.flame.burner.setAbsorbtionFactor(index,1.0)

self.flame.outlet.setSpeciesBcType(index,self.flame.burner.BC_VALUE)

    def setFlame(self, new_flame):
        """
        Used to replace the existing flame object with an externally
produced flame object.
        Purpose is for grid refinement study.

        new_flame = new BurnerFlame or BurnerFlame_efield object
        """
        self.flame = new_flame
        self.setOptionalProperties()
        self.setElectronModel()
        self.setElectricField()
        self.initFlame()
```

```
    def createFlame(self):
        """
        Used to create a new flame object
        """
        #create gas object
        self.gas = IdealGasMix(self.rxmech, self.rxmix)
        #create BurnerFlame or BurnerFlame_efield object
        if self.use_efield == True:
            self.flame = BurnerFlame_efield(gas = self.gas, grid =
self.initial_grid)
        else:
            self.flame = BurnerFlame(gas = self.gas, grid =
self.initial_grid)
        self.flame.gas.setState_TPY(self.temperature, self.pressure,
self.mass_fractions)
        mole_fractions = 'N2:' + str(self.flame.gas.moleFraction('N2'))
+ ',O2:' + str(self.flame.gas.moleFraction('O2')) + ',CH4:' +
str(self.flame.gas.moleFraction('CH4'))
        self.flame.burner.set(massflux = self.flow_rate, mole_fractions
= mole_fractions, temperature = self.temperature)

    def setOptionalProperties(self):
        """
        Sets up grid refinement properties
        """
        #optional properties
        if self.max_jacobian != None:
            self.flame.setMaxJacAge (self.max_jacobian[0],
self.max_jacobian[1])
        if (self.time_step != None and self.time_refine != None):
            self.flame.setTimeStep(self.time_step, self.time_refine)
        if (self.refine_ratio != None and self.refine_slope != None and
self.refine_curve != None):
            self.flame.setRefineCriteria(ratio = self.refine_ratio,
slope = self.refine_slope, curve = self.refine_curve)
        if (self.tol_ss != None and self.tol_ts != None):
            self.flame.set(tol = self.tol_ss, tol_time = self.tol_ts)

    def setElectricField(self):
        """
        Initial setup of electric field in flame object
        """
        #set the electric field
        if self.use_efield == True:
            self.flame.burner.setScalars( Numeric.array((0.0, 0.0,),
'd'))
            self.flame.outlet.setScalars(
Numeric.array((self.applied_voltage, 0.0), 'd'))
            self.flame.enableNeutralBC()

    def initFlame(self):
        """
        Initializes a flame object
        """
        #initialize
        if self.init_param == None:
            self.flame.init()
```

```
        else:
            self.flame.init(self.init_param)
        self.initialized = True

    def presolve (self):
        """
        This function is used to solve with no energy equations.  This
method is used to prepare the flame
        numerically for a full solution.
        """
        if self.initialized == True:
            self.flame.set(energy = 'off')
            self.flame.showSolution()
            try:
                self.flame.solve(self.log_level)
            except:
                print "Failure solving without energy!!"
        else:
            print "Flame must be initialized first!!"

    def solve (self, xml_dir = None, csv_dir = None):
        """
        This function attempts to solve the the case previously
assembled by other functions in this class.
        If desired, the output may be saved, but a directory strings
are required for saving.

        xml_dir = directory in which to save the resultant xml file.
        cvs_dir = directory in which to store the resultant cvs file

        The directory should simply be a string of the directory path
without a final '/'.  The file will be saved
        the form:

        'V_(voltage)_mtot_(total mass flux)_mair_(mass flux of
air)_mfuel_(mass flux of fuel).(xml or csv)'

        If the efield is not used, then the voltage will be dropped
from the filename.
        """
        if self.initialized == True:
            self.flame.set(energy = 'on')
            self.flame.solve(self.log_level)

            f_name = self.name_gen.createNameMethane(m_dot_total =
self.m_dot_total, m_dot_air = self.m_dot_air, m_dot_fuel =
self.m_dot_fuel, applied_voltage = self.applied_voltage)

            if xml_dir != None:
                self.flame.save(xml_dir + '/' + f_name + '.xml')
                print "XML file saved as " + xml_dir + '/' + f_name +
'.xml'

            if csv_dir != None:
                if self.use_efield == True:
                    z = self.flame.flame.grid()
                    T = self.flame.T()
```

```
                    u = self.flame.u()
                    V = self.flame.V()
                    Volt = self.flame.phi()
                    fcsv = open(csv_dir + '/' + f_name + '.csv' , 'w')
                    writeCSV(fcsv, ['Z (m)', 'U(m/s)', 'V (1/s)', 'T
(K)', 'Voltage'] + list(self.gas.speciesNames()))
                    for n in range(self.flame.flame.nPoints()):
                        self.flame.setGasState(n)
                        writeCSV(fcsv, [z[n], u[n], V[n], T[n],
Volt[n]] + list(self.gas.moleFractions()))
                    fcsv.close()
                    print "CSV written to " + csv_dir + '/' + f_name +
'.csv'
                    if self.current_file != None:
                        self.current_file.write(str(self.phi) + ',' +
str(self.m_dot_total) + ',' + str(self.m_dot_air) + ',' +
str(self.m_dot_fuel) + ',' + str(self.applied_voltage) + ',' +
str(self.flame.burner.current())+ '\n') #output info to current file
                        self.current_file.flush() #force file write now
                else:
                    z = self.flame.flame.grid()
                    T = self.flame.T()
                    u = self.flame.u()
                    V = self.flame.V()
                    fcsv = open(csv_dir + '/' + f_name + '.csv' , 'w')
                    writeCSV(fcsv, ['Z (m)', 'U(m/s)', 'V (1/s)', 'T
(K)'] + list(self.gas.speciesNames()))
                    for n in range(self.flame.flame.nPoints()):
                        self.flame.setGasState(n)
                        writeCSV(fcsv, [z[n], u[n], V[n], T[n]] +
list(self.gas.moleFractions()))
                    fcsv.close()
                    print 'CSV written to ' + csv_dir + '/' + f_name +
'.csv'
        else:
            print "Flame must be initialized before it can be solved!!"

    def change_voltage(self, new_volt):
        """
        Function used to change the voltage for an existing burner
object
        """
        if self.use_efield == True:
            self.applied_voltage = new_volt
            self.flame.outlet.setScalars(
Numeric.array((self.applied_voltage, 0.0), 'd'))
        else:
            print "Initialized as BurnerFlame, not BurnerFlame_efield.
Voltage change will have no affect."

    def __str__(self):
        """
        This function is used to print all class variables.
        """
        #Constant
        print " "
        print "Constants = "
```

```
        print "N2_PERCENT = " + str(self.N2_PERCENT)
        print "O2_PERCENT = " + str(self.O2_PERCENT)
        print "SAFR = " + str(self.SAFR)
        print "O2_MW = " + str(self.O2_MW)
        print "N2_MW = " + str(self.N2_MW)
        print "FUEL_MW = " + str(self.FUEL_MW)

        #Variable
        print " "
        print "Variable"
        print "applied_voltage = " + str(self.applied_voltage)
        print "area = " + str(self.area)
        print "flame = " + str(self.flame)
        print "flow_rate = " + str(self.flow_rate)
        print "gas = " + str(self.gas)
        print "initial_grid = " + str(self.initial_grid)
        print "log_level = " + str(self.log_level)
        print "max_jacobian = " + str(self.max_jacobian)
        print "m_dot_air = " + str(self.m_dot_air)
        print "m_dot_fuel = " + str(self.m_dot_fuel)
        print "m_dot_total = " + str(self.m_dot_total)
        print "mass_fractions = " + str(self.mass_fractions)
        print "phi = " + str(self.phi)
        print "pressure = " + str(self.pressure)
        print "refine_curve = " + str(self.refine_curve)
        print "refine_grid = " + str(self.refine_grid)
        print "refine_ratio = " + str(self.refine_ratio)
        print "refine_slope = " + str(self.refine_slope)
        print "rxmech = " + str(self.rxmech)
        print "rxmix = " + str(self.rxmix)
        print "time_step = " + str(self.time_step)
        print "time_refine = " + str(self.time_refine)
        print "tol_ss = " + str(self.tol_ss)
        print "tol_ts = " + str(self.tol_ts)
        print "temperature = " + str(self.temperature)
        print "use_efield = " + str(self.use_efield)

class EFSim_NG_Current_mass (EFSim_NG_Current):
    """
    Implements all functions provided by EFSim_NG, but the constructor
needs the Mass Flux of Air and Fuel
    seperately rather than the Total Mass Flux and Equivalence Ratio.
    """

    def __init__(self, m_dot_air, m_dot_fuel, area, rxmech, rxmix,
use_efield = False, applied_voltage = None, current_file = None):
        """
        Constructor used to generate EFSim object.
        Based on air and fuel flow rates.

        m_dot_air = mass flux of air
        m_dot_fuel = mass flux of fuel
        area = burner area in m^2
        rxmech = reaction mechanism file
        rxmix = mix type
        use_efield = True -- use Cantera.BurnerFlame_efield
                     False -- use Cantera.BurnerFlame
```

```
        applied_voltage = applied voltage
        current_file = name of file to store currents
        """
        #setup class data
        self.area = area
        self.m_dot_air = m_dot_air
        self.m_dot_fuel = m_dot_fuel
        self.m_dot_total = m_dot_air + m_dot_fuel
        afr = m_dot_air / m_dot_fuel
        self.phi = self.SAFR/afr
        self.flow_rate = self.m_dot_total / area
        self.use_efield = use_efield
        self.applied_voltage = applied_voltage
        self.rxmech = rxmech
        self.rxmix = rxmix

        #determine mix

        m_dot_n2 = self.N2_PERCENT * self.m_dot_air
        m_dot_o2 = self.O2_PERCENT * self.m_dot_air

        self.mass_fractions = "N2:" + str(m_dot_n2) + ",O2:" +
str(m_dot_o2) + ",CH4:" + str(self.m_dot_fuel)

        #test for erroneous use of efield
        if (applied_voltage == None and use_efield == True):
            print "Warning, Applied Voltage declared, but
BurnerFlame_efield is not selected for use."

        if current_file != None:
            self.current_file = open(current_file, 'w')

self.current_file.write('ER,total_flux,air_flux,fuel_flux,voltage,curre
nt\n')
        self.name_gen = np.nameParse()
```

A-2: Synthesis Gas Combustion Driver Code

Description:

The `EFSim_Syngas_Current` class has been constructed in order to allow for simplified execution and less computationally costly runs of multiple similar simulations for synthesis gas combustion. As with the `EFSim_NG_Current` class, one major purpose of this class is to allow for the reuse of flame objects for different simulations in which the only parameter to change is the applied voltage. The flame object can only be reliably reused if the major properties of the flame such as flame position, temperature profile, and profiles of major species remain approximately the same. These assumptions can be made because the voltages applied in these simulations are not high enough to create a significant ionic wind and the affected species make up a miniscule fraction of the total gas mixture. In order to include ionic wind effects, an additional governing equation would be required for momentum, but the effect is not strong enough to justify the required additional calculations.

Code Overview:

All of the code used to streamline synthesis gas combustion simulation is included the `EFSim_Syngas_Current` class. This class contains a single constructor which is used to create a simulation object, and the inputs to this constructor are given in the table below

Input Name	Purpose	Units
m_dot_air	Mass flux of air in the synthesis gas and air mix.	kg/s
m_dot_co	Mass flux of Carbon Monoxide in the synthesis gas and air mix.	kg/s
m_dot_h2	Mass flux of Hydrogen gas in the synthesis gas and air mix.	kg/s
m_dot_n2	Mass flux of Nitrogen gas from fuel in the synthesis gas and air mix.	kg/s
m_dot_ch4	Mass flux of Methane added to synthesis gas in the synthesis gas and air mix.	kg/s
area	Burner area of simulated flame object	m^2
rxmech	Filename of the Cantera "cti" file containing the desired reaction mechanism file.	---
rxmix	Name of the mix type within the reaction mechanism file to use for the simulation.	---
use_efield	Boolean flag used to denote if an applied electric field will be used in the desired simulation.	---
applied_voltage	Initial applied voltage for the simulation object.	V
current_file	Name of file to use for current calculation performed while running the flame object.	---

Table A-4: Constructor Inputs for Synthesis Gas Combustion Modeling Code

Within the EFSim_NG_Current class includes a set of "constants." Because Python does not allow for easy inclusion of Constant values, it is up the user not to manipulate these values.

Constant Name	Purpose	Value
MW_CO	Molecular Weight of CO	28.0101
MW_H2	Molecular Weight of H_2	2.01588
MW_N2	Molecular weight of N_2	28.0134
MW_CH4	Molecular weight of CH_4	16.04246
MW_O2	Molecular weight of O_2	31.9998
N2_PERCENT	Percent by mass of N_2 in air	0.76708
O2_PERCENT	Percent by mass of O_2 in air	0.23292

Table A-5: Constants Included in the `EFSim_Syngas_Current` Class

The remainder of the functions used in synthesis gas simulations are given in the table below.

Function	Purpose	Input	Description	Units
__init__	Class constructor	*see Table A-4		
calcSAFR	Calculates the stoichiometric air-fuel ratio for a methane doped synthesis gas mix	co_mass	Mass fraction of carbon monoxide in synthesis gas	---
		h2_mass	Mass fraction of hydrogen gas in synthesis gas	---
		n2_mass	Mass fraction of nitrogen gas in synthesis gas mixture	---
		ch4_mass	Mass fraction of methane in doped synthesis gas mixture	---
__del__	Class destructor used to close all open file before disposing of simulation object	---	---	---
initialize	Used to initialize a BurnerFlame or BurnerFlame_efield object for use in simulation	pressure	Gas pressure to use in the simulation	Pascal
		temperature	Burner temperature to use in the simulation	K
		initial_grid	List of the initial set of grid points to use in the simulation.	m
		refine_grid	Indicates if a simulation should attempt to add grid points as needed. 1 indicates that points may be added, and 0 indicates that points should not be added	---
		log_level	Indicates the level of diagnostic output to produce during a simulation. 1 is the minimum, and 5 is the maximum	---
		tol_ss	Steady state error tolerance.	---

		tol_ts	Transient state error tolerance	---
		max_jacobian	Used to limit the reuse of jacobian matrices during calculations	---
		refine_ratio	Ratio used to determine a need for solution refinement	---
		refine_slope	Slope used to determine a need for solution refinement	---
		refine_curve	Curve used to determine a need for solution refinement	---
		time_step	Length of time steps to use while finding steady state solutions	s
		time_refine	List of number of timesteps to take at a time while finding a solution	---
		electron_mobility	Electron mobility to be used in a simulation	m^2/Vs
		init_param	Normalized starting flame position used to aid in flame calculation by allowing an improved initial guess.	---
setElectronModel	Function used to set the electron mobility within the flame object	---	---	---
setFlame	Function used to allow for the use of an externally created BurnerFlame or BurnerFlame_efield object for a simulation.	new_flame	The BurnerFlame or BurnerFlame_efield object to use in a simulation.	---
createFlame	Function intended for internal use in creating a	---	---	---

	BurnerFlame or BurnerFlame_efield object			
setOptionalProperties	Function intended for internal use in setting optional properties given as inputs to the initialize function	---	---	---
setElectricField	Function intended for internal use in initially setting up an electric field for the flame object	---	---	---
initFlame	Used to call Cantera's initialize function on the flame object and flag the flame object as initialized	---	---	---
presolve	This function solves the flame object with the energy equations turned off in order to derive a better initial solution before running a true simulation	---	---	---
solve	This function solves the flame object with the energy equation turned on and saves the resulting output files.	xml_dir	Name of directory in which to store results in XML file format	---
		csv_dir	Name of directory in which to store results in CSV file format	---
change_voltage	Function used to change the voltage between runs of similar simulations	new_volt	Value for the new applied voltage	V
__str__	Function used to display class constants and variables	---	---	---

Table A-6: Class Methods for `EFSim_Syngas_Current Class`

As a result of running simulations with these programs, three different types of output files are generated. The first type is a solution in Cantera's 'ctml' file format. The second type is also a solution, but this solution is in comma separated value ('csv') format. The third potential output file is a file listing the currents for each simulation run. All three of these files formats are documented in Appendix D. The names for the first two files are generated using the nameParse class, which is specified in Appendix C.

Code Listing:

```
from Cantera import *
from Cantera.OneD import *
from Numeric import *
import nameParse as np

class EFSim_Syngas_Current:
    """
    This class holds data and methods for a single flame simulation.
The constructor for this class uses Total
    Mass Flux and Equivalence Ratio.  To use Fuel and Air Mass Flux
seperately, use the EFSim_Syngas_Current_mass class instead.
    """
    #Constant
    MW_CO = 28.0101
    MW_H2 = 2.01588
    MW_N2 = 28.0134
    MW_CH4 = 16.04246
    MW_O2 = 31.9988
    N2_PERCENT = 0.76708 # N2 percent by mass in air
    O2_PERCENT = 0.23292 # O2 percent by mass in air

    #Variable
    applied_voltage = None
    area = None
    flame = None
    flow_rate = None
    gas = None
    initial_grid = None
    log_level = None
    max_jacobian = None
    m_dot_air = None
    m_dot_co = None
    m_dot_ch4 = None
    m_dot_h2 = None
    m_dot_n2 = None
    m_dot_total = None
    mass_fractions = None
    phi = None
    pressure = None
    refine_curve = None
```

```
    refine_grid = None
    refine_ratio = None
    refine_slope = None
    rxmech = None
    rxmix = None
    safr = None
    time_step = None
    time_refine = None
    tol_ss = None
    tol_ts = None
    temperature = None
    use_efield = None
    current_file = None
    modelFactor = None
    name_gen = None
    init_param = None

    def __init__(self, m_dot_air, m_dot_co, m_dot_h2, m_dot_n2,
m_dot_ch4, area, rxmech, rxmix, use_efield = False, applied_voltage =
None, current_file = None):
        """
        Constructor used to generate EFSim_Syngas_Current object.
        Based on fuel mixture, air and fuel flow rates.

        m_dot_air = mass flux of air
        m_dot_co = mass flux of Carbon Monoxide
        m_dot_h2 = mass flux of Hydrogen gas
        m_dot_n2 = mass flux of Nitrogen gas in the fuel mixture
        m_dot_ch4 = mass flux of Methane in the fuel mixture
        area = burner area in m^2
        rxmech = reaction mechanism file
        rxmix = mix type
        use_efield = True -- use Cantera.BurnerFlame_efield
                     False -- use Cantera.BurnerFlame
        applied_voltage = applied voltage
        current_file = name of file to store currents
        """
        #setup class data
        self.area = area
        self.m_dot_air = m_dot_air
        self.m_dot_co = m_dot_co
        self.m_dot_h2 = m_dot_h2
        self.m_dot_n2 = m_dot_n2
        self.m_dot_ch4 = m_dot_ch4
        self.m_dot_total = m_dot_air + m_dot_co + m_dot_h2 + m_dot_n2 +
m_dot_ch4
        self.flow_rate = self.m_dot_total / self.area
        self.use_efield = use_efield
        self.applied_voltage = applied_voltage
        self.rxmech = rxmech
        self.rxmix = rxmix

        #get Equivalence Ratio
        m_dot_fuel_tot = m_dot_co + m_dot_h2 + m_dot_n2 + m_dot_ch4
        co_per = m_dot_co/m_dot_fuel_tot
        h2_per = m_dot_h2/m_dot_fuel_tot
        n2_per = m_dot_n2/m_dot_fuel_tot
```

```
        ch4_per = m_dot_ch4/m_dot_fuel_tot
        self.safr = self.calcSAFR(co_per, h2_per, n2_per, ch4_per)
        afr = m_dot_air / (m_dot_co + m_dot_h2 + m_dot_n2 + m_dot_ch4)
# bug fixed
        self.phi = self.safr/afr

        #determine mix
        m_dot_air_n2 = self.N2_PERCENT * self.m_dot_air
        m_dot_air_o2 = self.O2_PERCENT * self.m_dot_air

        self.mass_fractions = "N2:" + str(m_dot_n2 + m_dot_air_n2) +
",O2:" + str(m_dot_air_o2) + ",CO:" + str(self.m_dot_co) + ",H2:" +
str(m_dot_h2) + ",CH4:" + str(self.m_dot_ch4)

        #test for erroneous use of efield
        if (applied_voltage == None and use_efield == True):
            print "Warning, Applied Voltage declared, but
BurnerFlame_efield is not selected for use."

        if current_file != None:
            self.current_file = open(current_file, 'w')

self.current_file.write('ER,total_flux,air_flux,CO_flux,H2_flux,CH4_flu
x,N2_flux,voltage,current\n')

        self.name_gen = np.nameParse()

    def calcSAFR(self, co_mass, h2_mass, n2_mass, ch4_mass):
        """
        This function calculates a value for the Stoichiometric
        Air/Fuel Ratio(in Mass) for syngas doped with methane given
        the mass percentages of CO, H2, N2, and CH4

        inputs:
        co_mass = mass fraction of CO
        h2_mass = mass fraction of H2
        n2_mass = mass fraction of N2
        ch4_mass = mass fraction of CH4

        outputs:
        SAFR = Stochiometric Air/Fuel Ratio for the specified input mix
        """
        # determine mole fractions from mass fractions
        co_mole = co_mass/self.MW_CO
        h2_mole = h2_mass/self.MW_H2
        n2_mole = n2_mass/self.MW_N2
        ch4_mole = ch4_mass/self.MW_CH4
        mole_tot = co_mole + h2_mole + n2_mole + ch4_mole
        co_mole = co_mole/mole_tot
        h2_mole = h2_mole/mole_tot
        n2_mole = n2_mole/mole_tot
        ch4_mole = ch4_mole/mole_tot

        # determine air for stoichiometric mix
        mole_air = co_mole * 0.5 + h2_mole*0.5 + ch4_mole*2.0
        mass_air = mole_air * self.MW_O2 + mole_air * 3.76 * self.MW_N2
```

```
        mass_fuel = co_mole * self.MW_CO + h2_mole * self.MW_H2 +
n2_mole * self.MW_N2 + ch4_mole * self.MW_CH4
        SAFR = mass_air/mass_fuel
        return SAFR


    def __del__(self):
        """
        Destructor for EFSim_Syngas_Current.
        """
        if self.current_file != None:
            self.current_file.close()

    def initialize (self, pressure, temperature, initial_grid,
                    refine_grid = 1, log_level = 5,
                    tol_ss = None, tol_ts = None,
                    max_jacobian = None,
                    refine_ratio = None, refine_slope = None,
                    refine_curve = None,
                    time_step = None, time_refine = None,
                    electron_mobility = 0.4, init_param = None):
        """
        This function is used to initialize a BurnerFlame or
BurnerFlame_efield.
        Inputs as provided to Cantera are required.

        pressure = gas pressure
        temperature = burner temperature in Kelvin
        initial grid = list of points to start evaluation
        refine_grid = 1 to refine the grid when solvin, 0 otherwise
        log_level = level of detail in cantera output, 1 is the
minimum, 5 is the maximum
        tol_ss = steady state tolerance (list of 2 values)
        tol_ts = transient state tolerance (list of 2 values)
        max_jacobian = maximum jacobian ages.  (tuple of 2 values)
        refine_ratio = ratio to adapt refine criteria
        refine_slope = slope to adapt refine criteria
        refine_curve = curve to adapt refine criteria
        time_step = time in seconds for a single timestep
        time_refine = list of number of timesteps to use when refining
        electron_mobility = constant value for the electron mobility
                            -- default value = 0.4
        init_param = normalized starting flame position

        pressure, temperature, and initial_grid must be set.
        All other attributes will be set to default values if not
specified.
        In most cases, the default value is the Cantera default, but
for electron
        mobility, the default value is 0.4.
        """

        #store attributes
        self.pressure = pressure
        self.temperature = temperature
        self.initial_grid = initial_grid
        self.refine_grid = refine_grid
```

```
        self.log_level = log_level
        self.tol_ss = tol_ss
        self.tol_ts = tol_ts
        self.max_jacobian = max_jacobian
        self.refine_ratio = refine_ratio
        self.refine_slope = refine_slope
        self.refine_curve = refine_curve
        self.time_step = time_step
        self.time_refine = time_refine
        self.modelFactor = electron_mobility
        self.init_param = init_param

        #perform setup
        self.createFlame()
        self.setOptionalProperties()
        self.setElectronModel()
        self.setElectricField()
        self.initFlame()

    def setElectronModel(self):
        """
        This function is used to set the electron model of all charged
species.
        High probability for change.
        """
        if self.use_efield == True:
            electronModel = self.flame.flame.emodel_CONST
            useMassGradient = 0
            self.flame.flame.setElectronTransModel(electronModel,
useMassGradient, self.modelFactor)
            for index in range(0,self.flame.gas.nSpecies()):
                if self.flame.gas.nAtoms(index,'E') < 0:
                    self.flame.burner.setAbsorbtionFactor(index,1.0)

self.flame.burner.setSpeciesBcType(index,self.flame.burner.BC_THERMAL)
### Perhaps BC_Value

self.flame.outlet.setSpeciesBcType(index,self.flame.burner.BC_VALUE)
                elif self.flame.gas.nAtoms(index,'E') > 0:
                    self.flame.burner.setSpeciesBcType(index,
self.flame.burner.BC_VALUE)
                    self.flame.burner.setAbsorbtionFactor(index,1.0)

self.flame.outlet.setSpeciesBcType(index,self.flame.burner.BC_VALUE)

    def setFlame(self, new_flame):
        """
        Used to replace the existing flame object with an externally
produced flame object.
        Purpose is for grid refinement study.
        new_flame = new BurnerFlame or BurnerFlame_efield object
        """
        self.flame = new_flame
        self.setOptionalProperties()
        self.setElectronModel()
        self.setElectricField()
        self.initFlame()
```

```
    def createFlame(self):
        """
        Used to create a new flame object
        """
        #create gas object
        self.gas = IdealGasMix(self.rxmech, self.rxmix)
        #create BurnerFlame or BurnerFlame_efield object
        if self.use_efield == True:
            self.flame = BurnerFlame_efield(gas = self.gas, grid =
self.initial_grid)
        else:
            self.flame = BurnerFlame(gas = self.gas, grid =
self.initial_grid)
        self.flame.gas.setState_TPY(self.temperature, self.pressure,
self.mass_fractions)
        mole_fractions = ""
        for species in ('N2', 'O2', 'CH4', 'H2', 'CO'):
            mole_fractions += species + ':' +
str(self.flame.gas.moleFraction(species)) + ','
        mole_fractions = mole_fractions[:-2]
        self.flame.burner.set(massflux = self.flow_rate, mole_fractions
= mole_fractions, temperature = self.temperature)

    def setOptionalProperties(self):
        """
        Sets up grid refinement properties
        """
        #optional properties
        if self.max_jacobian != None:
            self.flame.setMaxJacAge (self.max_jacobian[0],
self.max_jacobian[1])
        if (self.time_step != None and self.time_refine != None):
            self.flame.setTimeStep(self.time_step, self.time_refine)
        if (self.refine_ratio != None and self.refine_slope != None and
self.refine_curve != None):
            self.flame.setRefineCriteria(ratio = self.refine_ratio,
slope = self.refine_slope, curve = self.refine_curve)
        if (self.tol_ss != None and self.tol_ts != None):
            self.flame.set(tol = self.tol_ss, tol_time = self.tol_ts)

    def setElectricField(self):
        """
        Initial setup of electric field in flame object
        """
        #set the electric field
        if self.use_efield == True:
            self.flame.burner.setScalars( Numeric.array((0.0, 0.0,),
'd'))
            self.flame.outlet.setScalars(
Numeric.array((self.applied_voltage, 0.0), 'd'))
            self.flame.enableNeutralBC()

    def initFlame(self):
        """
        Initializes a flame object
        """
```

```
        #initialize
        if self.init_param == None:
            self.flame.init()
        else:
            self.flame.init(self.init_param)
        self.initialized = True

    def presolve (self):
        """
        This function is used to solve with no energy equations.  This
method is used to prepare the flame
        numerically for a full solution.
        """
        if self.initialized == True:
            self.flame.set(energy = 'off')
            self.flame.showSolution()
            try:
                self.flame.solve(self.log_level)
            except:
                print "Failure solving without energy!!"
        else:
            print "Flame must be initialized first!!"

    def solve (self, xml_dir = None, csv_dir = None):
        """
        This function attempts to solve the the case previously
assembled by other functions in this class.
        If desired, the output may be saved, but a directory strings
are required for saving.

        xml_dir = directory in which to save the resultant xml file.
        cvs_dir = directory in which to store the resultant cvs file

        The directory should simply be a string of the directory path
without a final '/'.  The file will be saved
        the form:

        'V_(voltage)_mtot_(total mass flux)_mair_(mass flux of
air)_mfuel_(mass flux of fuel).(xml or csv)'

        If the efield is not used, then the voltage will be dropped
from the filename.
        """
        if self.initialized == True:
            self.flame.set(energy = 'on')
            self.flame.solve(self.log_level)
            f_name = self.name_gen.createNameSyngas(m_dot_total =
self.m_dot_total, m_dot_air = self.m_dot_air, m_dot_co = self.m_dot_co,
                    m_dot_h2 = self.m_dot_h2, m_dot_n2 =
self.m_dot_n2, m_dot_ch4 = self.m_dot_ch4, applied_voltage =
self.applied_voltage)
            if xml_dir != None:
                self.flame.save(xml_dir + '/' + f_name + '.xml')
                print "XML file saved as " + xml_dir + '/' + f_name +
'.xml'
            if csv_dir != None:
                if self.use_efield == True:
```

```
                    z = self.flame.flame.grid()
                    T = self.flame.T()
                    u = self.flame.u()
                    V = self.flame.V()
                    Volt = self.flame.phi()
                    fcsv = open(csv_dir + '/' + f_name + '.csv' , 'w')
                    writeCSV(fcsv, ['Z (m)', 'U(m/s)', 'V (1/s)', 'T
(K)', 'Voltage'] + list(self.gas.speciesNames()))
                    for n in range(self.flame.flame.nPoints()):
                        self.flame.setGasState(n)
                        writeCSV(fcsv, [z[n], u[n], V[n], T[n],
Volt[n]] + list(self.gas.moleFractions()))
                    fcsv.close()
                    print "CSV written to " + csv_dir + '/' + f_name +
'.csv'
                    if self.current_file != None:
                        self.current_file.write(str(self.phi) + ',' +
str(self.m_dot_total) + ',' + str(self.m_dot_air) + ',' +
str(self.m_dot_co) + ',' + str(self.m_dot_h2) + ',' +
str(self.m_dot_ch4) + ',' + str(self.m_dot_n2) + ',' +
str(self.applied_voltage) + ',' + str(self.flame.burner.current())+
'\n') #output info to current file
                        self.current_file.flush() #force file write now
                else:
                    z = self.flame.flame.grid()
                    T = self.flame.T()
                    u = self.flame.u()
                    V = self.flame.V()
                    fcsv = open(csv_dir + '/' + f_name + '.csv' , 'w')
                    writeCSV(fcsv, ['Z (m)', 'U(m/s)', 'V (1/s)', 'T
(K)'] + list(self.gas.speciesNames()))
                    for n in range(self.flame.flame.nPoints()):
                        self.flame.setGasState(n)
                        writeCSV(fcsv, [z[n], u[n], V[n], T[n]] +
list(self.gas.moleFractions()))
                    fcsv.close()
                    print 'CSV written to ' + csv_dir + '/' + f_name +
'.csv'
        else:
            print "Flame must be initialized before it can be solved!!"

    def change_voltage(self, new_volt):
        """
        Function used to change the voltage for an existing burner
object
        """
        if self.use_efield == True:
            self.applied_voltage = new_volt
            self.flame.outlet.setScalars(
Numeric.array((self.applied_voltage, 0.0), 'd'))
        else:
            print "Initialized as BurnerFlame, not BurnerFlame_efield.
Voltage change will have no affect."

    def __str__(self): #fix me upon completion
        """
        This function is used to print all class variables.
```

```
"""
#Constant
print " "
print "Constants"
print "N2_PERCENT = " + str(self.N2_PERCENT)
print "O2_PERCENT = " + str(self.O2_PERCENT)
print "MW_CH4 = " + str(self.MW_CH4)
print "MW_CO = " + str(self.MW_CO)
print "MW_H2 = " + str(self.MW_H2)
print "MW_O2 = " + str(self.MW_O2)
print "MW_N2 = " + str(self.MW_O2)

#Variable
print " "
print "Variable"
print "applied_voltage = " + str(self.applied_voltage)
print "area = " + str(self.area)
print "flame = " + str(self.flame)
print "flow_rate = " + str(self.flow_rate)
print "gas = " + str(self.gas)
print "initial_grid = " + str(self.initial_grid)
print "log_level = " + str(self.log_level)
print "max_jacobian = " + str(self.max_jacobian)
print "m_dot_air = " + str(self.m_dot_air)
print "m_dot_co = " + str(self.m_dot_co)
print "m_dot_ch4 = " + str(self.m_dot_ch4)
print "m_dot_h2 = " + str(self.m_dot_h2)
print "m_dot_n2 = " + str(self.m_dot_n2)
print "m_dot_total = " + str(self.m_dot_total)
print "mass_fractions = " + str(self.mass_fractions)
print "model_factor = " + str(self.model_factor)
print "phi = " + str(self.phi)
print "pressure = " + str(self.pressure)
print "refine_curve = " + str(self.refine_curve)
print "refine_grid = " + str(self.refine_grid)
print "refine_ratio = " + str(self.refine_ratio)
print "refine_slope = " + str(self.refine_slope)
print "rxmech = " + str(self.rxmech)
print "rxmix = " + str(self.rxmix)
print "time_step = " + str(self.time_step)
print "time_refine = " + str(self.time_refine)
print "tol_ss = " + str(self.tol_ss)
print "tol_ts = " + str(self.tol_ts)
print "temperature = " + str(self.temperature)
print "use_efield = " + str(self.use_efield)
```

Appendix B: Cantera Reaction Mechanism Files

B-1: Listing of Jones, Heinsohn and Becker Mechanism

```
#
# Generated from file methane_ion31.che
# by ck2cti on Tue Apr 27 11:38:05 2004
#
# Transport data from file methion31trans.dat.

units(length = "cm", time = "s", quantity = "mol", act_energy =
"cal/mol")


ideal_gas(name = "gas",
      elements = " H  O  C  N  E ",
      species = """ CH4  O2  CO2  H2O  CO  H2  OH  CH3  CH2O  CH
                  H  O  E  HCO+  H3O+  N  NO  NO2  N2 """,
      reactions = "all",
      transport = "Mix",
      initial_state = state(temperature = 300.0,
                        pressure = OneAtm)    )



#-----------------------------------------------------------------------
---------
#  Species data
#-----------------------------------------------------------------------
---------

species(name = "CH4",
    atoms = " C:1  H:4 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  5.149876130E+00,  -1.367097880E-
02,
                4.918005990E-05,  -4.847430260E-08,   1.666939560E-11,
               -1.024664760E+04,  -4.641303760E+00] ),
       NASA( [ 1000.00,  3500.00], [  7.485149500E-02,   1.339094670E-
02,
               -5.732858090E-06,   1.222925350E-09,  -1.018152300E-13,
               -9.468344590E+03,   1.843731800E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.75,
                     well_depth =   141.40,
                     polar =     2.60,
                     rot_relax =    13.00),
    note = "L 8/88"
       )

species(name = "O2",
    atoms = " O:2 ",
```

```
    thermo = (
       NASA( [  200.00,  1000.00], [  3.782456360E+00,  -2.996734160E-
03,
                9.847302010E-06,  -9.681295090E-09,   3.243728370E-12,
               -1.063943560E+03,   3.657675730E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.282537840E+00,   1.483087540E-
03,
               -7.579666690E-07,   2.094705550E-10,  -2.167177940E-14,
               -1.088457720E+03,   5.453231290E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.46,
                     well_depth =   107.40,
                     polar =     1.60,
                     rot_relax =     3.80),
    note = "TPIS89"
       )

species(name = "CO2",
    atoms = " C:1  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.356773520E+00,   8.984596770E-
03,
               -7.123562690E-06,   2.459190220E-09,  -1.436995480E-13,
               -4.837196970E+04,   9.901052220E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.857460290E+00,   4.414370260E-
03,
               -2.214814040E-06,   5.234901880E-10,  -4.720841640E-14,
               -4.875916600E+04,   2.271638060E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.76,
                     well_depth =   244.00,
                     polar =     2.65,
                     rot_relax =     2.10),
    note = "L 7/88"
       )

species(name = "H2O",
    atoms = " H:2  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.198640560E+00,  -2.036434100E-
03,
                6.520402110E-06,  -5.487970620E-09,   1.771978170E-12,
               -3.029372670E+04,  -8.490322080E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.033992490E+00,   2.176918040E-
03,
               -1.640725180E-07,  -9.704198700E-11,   1.682009920E-14,
               -3.000429710E+04,   4.966770100E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.60,
                     well_depth =   572.40,
                     dipole =     1.84,
```

```
                    rot_relax =     4.00),
    note = "L 8/89"
       )

species(name = "CO",
    atoms = " C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.579533470E+00,  -6.103536800E-
04,
                1.016814330E-06,   9.070058840E-10,  -9.044244990E-13,
               -1.434408600E+04,   3.508409280E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.715185610E+00,   2.062527430E-
03,
               -9.988257710E-07,   2.300530080E-10,  -2.036477160E-14,
               -1.415187240E+04,   7.818687720E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.65,
                    well_depth =    98.10,
                    polar =     1.95,
                    rot_relax =     1.80),
    note = "TPIS79"
       )

species(name = "H2",
    atoms = " H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.344331120E+00,   7.980520750E-
03,
               -1.947815100E-05,   2.015720940E-08,  -7.376117610E-12,
               -9.179351730E+02,   6.830102380E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.337279200E+00,  -4.940247310E-
05,
                4.994567780E-07,  -1.795663940E-10,   2.002553760E-14,
               -9.501589220E+02,  -3.205023310E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     2.92,
                    well_depth =    38.00,
                    polar =     0.79,
                    rot_relax =   280.00),
    note = "TPIS78"
       )

species(name = "OH",
    atoms = " O:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.992015430E+00,  -2.401317520E-
03,
                4.617938410E-06,  -3.881133330E-09,   1.364114700E-12,
                3.615080560E+03,  -1.039254580E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.092887670E+00,   5.484297160E-
04,
                1.265052280E-07,  -8.794615560E-11,   1.174123760E-14,
                3.858657000E+03,   4.476696100E+00] )
```

```
            ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     2.75,
                    well_depth =    80.00),
    note = "RUS 78"
       )

species(name = "CH3",
    atoms = " C:1  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.673590400E+00,   2.010951750E-
03,
                5.730218560E-06,  -6.871174250E-09,   2.543857340E-12,
                1.644499880E+04,   1.604564330E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.285717720E+00,   7.239900370E-
03,
               -2.987143480E-06,   5.956846440E-10,  -4.671543940E-14,
                1.677558430E+04,   8.480071790E+00] )
            ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.80,
                    well_depth =   144.00),
    note = "L11/89"
       )

species(name = "CH2O",
    atoms = " H:2  C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.793723150E+00,  -9.908333690E-
03,
                3.732200080E-05,  -3.792852610E-08,   1.317726520E-11,
               -1.430895670E+04,   6.028129000E-01] ),
       NASA( [ 1000.00,  3500.00], [  1.760690080E+00,   9.200000820E-
03,
               -4.422588130E-06,   1.006412120E-09,  -8.838556400E-14,
               -1.399583230E+04,   1.365632300E+01] )
            ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     3.59,
                    well_depth =   498.00,
                    rot_relax =     2.00),
    note = "L 8/88"
       )

species(name = "CH",
    atoms = " C:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.489816650E+00,   3.238355410E-
04,
               -1.688990650E-06,   3.162173270E-09,  -1.406090670E-12,
                7.079729340E+04,   2.084011080E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.878464730E+00,   9.709136810E-
04,
                1.444456550E-07,  -1.306878490E-10,   1.760793830E-14,
```

```
               7.101243640E+04,   5.484979990E+00] )
            ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "TPIS79"
       )

species(name = "H",
    atoms = " H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.500000000E+00,   7.053328190E-
13,
              -1.995919640E-15,   2.300816320E-18,  -9.277323320E-22,
               2.547365990E+04,  -4.466828530E-01] ),
       NASA( [ 1000.00,  3500.00], [  2.500000010E+00,  -2.308429730E-
11,
               1.615619480E-14,  -4.735152350E-18,   4.981973570E-22,
               2.547365990E+04,  -4.466829140E-01] )
            ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     2.05,
                     well_depth =   145.00),
    note = "L 7/88"
       )

species(name = "O",
    atoms = " O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.168267100E+00,  -3.279318840E-
03,
               6.643063960E-06,  -6.128066240E-09,   2.112659710E-12,
               2.912225920E+04,   2.051933460E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.569420780E+00,  -8.597411370E-
05,
               4.194845890E-08,  -1.001777990E-11,   1.228336910E-15,
               2.921757910E+04,   4.784338640E+00] )
            ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "L 1/90"
       )

species(name = "E",
    atoms = " E:1 ",
    thermo = (
       NASA( [  200.00,     5.49], [  2.500000000E+00,
0.000000000E+00,
               0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
              -7.453750000E+02,  -1.172469020E+01] ),
       NASA( [    5.49,  6000.00], [  2.500000000E+00,
0.000000000E+00,
               0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
```

```
               -7.453750000E+02,  -1.172469020E+01] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =   425.00,
                     well_depth =   850.00,
                     rot_relax =     1.00),
    note = "L10/92"
       )

species(name = "HCO+",
    atoms = " H:1  C:1  O:1  E:-1 ",
    thermo = (
       NASA( [  300.00,    29.02], [  2.473973600E+00,   8.671559000E-
03,
               -1.003150000E-05,   6.717052700E-09,  -1.787267400E-12,
                9.914660800E+04,   8.175711870E+00] ),
       NASA( [   29.02,  5000.00], [  3.741188000E+00,   3.344151700E-
03,
               -1.239712100E-06,   2.118938800E-10,  -1.370415000E-14,
                9.888407800E+04,   2.078613570E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00),
    note = "J12/70"
       )

species(name = "H3O+",
    atoms = " H:3  O:1  E:-1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.198640560E+00,  -2.036434100E-
03,
                6.520402110E-06,  -5.487970620E-09,   1.771978170E-12,
               -3.029372670E+04,  -8.490322080E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.033992490E+00,   2.176918040E-
03,
               -1.640725180E-07,  -9.704198700E-11,   1.682009920E-14,
               -3.000429710E+04,   4.966770100E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.60,
                     well_depth =   572.40,
                     dipole =     1.84,
                     rot_relax =     4.00),
    note = "L 8/89"
       )

species(name = "N",
    atoms = " N:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
                5.610463700E+04,   4.193908700E+00] ),
```

```
       NASA( [ 1000.00,  6000.00], [  2.415942900E+00,   1.748906500E-
04,
                -1.190236900E-07,   3.022624500E-11,  -2.036098200E-15,
                 5.613377300E+04,   4.649609600E+00] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     3.30,
                     well_depth =    71.40),
    note = "L 6/88"
       )

species(name = "NO",
    atoms = " N:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.218476300E+00,  -4.638976000E-
03,
                1.104102200E-05,  -9.336135400E-09,   2.803577000E-12,
                9.844623000E+03,   2.280846400E+00] ),
       NASA( [ 1000.00,  6000.00], [  3.260605600E+00,   1.191104300E-
03,
                -4.291704800E-07,   6.945766900E-11,  -4.033609900E-15,
                 9.920974600E+03,   6.369302700E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.62,
                     well_depth =    97.53,
                     polar =     1.76,
                     rot_relax =     4.00),
    note = "RUS 78"
       )

species(name = "NO2",
    atoms = " N:1  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.944031200E+00,  -1.585429000E-
03,
                1.665781200E-05,  -2.047542600E-08,   7.835056400E-12,
                2.896617900E+03,   6.311991700E+00] ),
       NASA( [ 1000.00,  6000.00], [  4.884754200E+00,   2.172395600E-
03,
                -8.280690600E-07,   1.574751000E-10,  -1.051089500E-14,
                 2.316498300E+03,  -1.174169500E-01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.50,
                     well_depth =   200.00,
                     rot_relax =     1.00),
    note = "L 7/88"
       )

species(name = "N2",
    atoms = " N:2 ",
    thermo = (
```

```
       NASA( [  300.00,  1000.00], [  3.298677000E+00,   1.408240400E-
03,
               -3.963222000E-06,   5.641515000E-09,  -2.444854000E-12,
               -1.020899900E+03,   3.950372000E+00] ),
       NASA( [ 1000.00,  5000.00], [  2.926640000E+00,   1.487976800E-
03,
               -5.684760000E-07,   1.009703800E-10,  -6.753351000E-15,
               -9.227977000E+02,   5.980528000E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.62,
                    well_depth =    97.53,
                    polar =     1.76,
                    rot_relax =     4.00),
    note = "121286"
       )



#-----------------------------------------------------------------------
---------
#  Reaction data
#-----------------------------------------------------------------------
---------

#  Reaction 1
reaction( "CH4 => CH3 + H",   [4.23000E+15, 0, 108690])

#  Reaction 2
reaction( "CH4 + OH => CH3 + H2O",   [2.00000E+14, 0, 7410])

#  Reaction 3
reaction( "CH4 + O => CH3 + OH",   [3.48000E+13, 0, 8380])

#  Reaction 4
reaction( "CH4 + H => CH3 + H2",   [4.35000E+14, 0, 13740])

#  Reaction 5
reaction( "CH3 + O2 => CH2O + OH",   [5.29000E+11, 0, 1700])

#  Reaction 6
reaction( "CH2O + OH => CO + H2O + H",   [5.87000E+14, 0, 4880])

#  Reaction 7
reaction( "CO + OH => CO2 + H",   [1.30000E+12, 0, 1530])

#  Reaction 8
reaction( "CO2 + H => CO + OH",   [1.45000E+14, 0, 23760])

#  Reaction 9
reaction( "O2 + H => OH + O",   [2.24000E+14, 0, 16800])

#  Reaction 10
reaction( "OH + O => O2 + H",   [1.71000E+13, 0, 870])

#  Reaction 11
```

```
reaction( "O + H2 => OH + H",   [1.74000E+13, 0, 9450])

#  Reaction 12
reaction( "OH + H => O + H2",   [7.70000E+12, 0, 7580])

#  Reaction 13
reaction( "O + H2O => 2 OH",   [5.75000E+13, 0, 18100])

#  Reaction 14
#  made a mistake earlier
reaction( "2 OH => O + H2O",   [5.38000E+12, 0, 1050])

#  Reaction 15
reaction( "OH + H2 => H2O + H",   [2.19000E+13, 0, 5150])

#  Reaction 16
reaction( "H2O + H => H2 + OH",   [8.41000E+13, 0, 20100])

#  Reaction 17
three_body_reaction( "H + OH + M => H2O + M",   [2.00000E+19, -1, 0])

#  Reaction 18
three_body_reaction( "O + O + M => O2 + M",   [8.90000E+14, -0.5, 0])

#  Reaction 19
three_body_reaction( "H + H + M => H2 + M",   [1.00000E+18, -1, 0])

#  Reaction 20
reaction( "CH3 + O => CH + H2O",   [2.80000E+08, 0, 0])

#  Reaction 21
reaction( "CH + O => HCO+ + E",   [5.75000E+11, 0, 6000])

#  Reaction 22
reaction( "HCO+ + H2O => CO + H3O+",   [5.02000E+17, 0, 24000])

#  Reaction 23
reaction( "H3O+ + E => H2O + H",   [1.44000E+17, 0, 0])

#  Reaction 24
reaction( "CH + O2 => CO + OH",   [6.00000E+10, 0, 0])

#  Reaction 25
reaction( "O + N2 => NO + N",   [1.36000E+14, 0, 75400])

#  Reaction 26
reaction( "NO + N => O + N2",   [3.10000E+13, 0, 330])

#  Reaction 27
reaction( "N + O2 => NO + O",   [6.43000E+09, 1, 6250])

#  Reaction 28
reaction( "NO + O => N + O2",   [1.55000E+09, 1, 38640])

#  Reaction 29
three_body_reaction( "NO + O + M => NO2 + M",   [1.05000E+15, 0, -
1870])
```

```
#  Reaction 30
reaction( "NO2 + O => NO + O2",   [2.10000E+12, 0, 0])

#  Reaction 31
reaction( "NO2 + H => NO + OH",   [3.00000E+14, 0, 0])
```

B-2: Listing of Modified GRI 3.0 Mechanism

```
#
# Generated from file grimech30.dat
# by ck2cti on Fri May 27 13:09:31 2005
#
# Transport data from file transport.dat.

units(length = "cm", time = "s", quantity = "mol", act_energy =
"cal/mol")


ideal_gas(name = "gas",
      elements = " O  H  C  N  Ar E ",
      species = """ H2  H  O  O2  OH  H2O  HO2  H2O2  C  CH
                    CH2  CH2(S)  CH3  CH4  CO  CO2  HCO  CH2O  CH2OH
CH3O
                    CH3OH  C2H  C2H2  C2H3  C2H4  C2H5  C2H6  HCCO
CH2CO  HCCOH
                    N  NH  NH2  NH3  NNH  NO  NO2  N2O  HNO  CN
                    HCN  H2CN  HCNN  HCNO  HOCN  HNCO  NCO  N2  AR  C3H7
                    C3H8  CH2CHO  CH3CHO E HCO+ H3O+ """,
      reactions = "all",
      transport = "Mix",
      initial_state = state(temperature = 300.0,
                        pressure = OneAtm)    )



#-----------------------------------------------------------------------
---------
#  Species data
#-----------------------------------------------------------------------
---------

species(name = "H2",
    atoms = " H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.344331120E+00,   7.980520750E-
03,
               -1.947815100E-05,   2.015720940E-08,  -7.376117610E-12,
               -9.179351730E+02,   6.830102380E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.337279200E+00,  -4.940247310E-
05,
                4.994567780E-07,  -1.795663940E-10,   2.002553760E-14,
               -9.501589220E+02,  -3.205023310E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.92,
                     well_depth =    38.00,
                     polar =     0.79,
                     rot_relax =   280.00),
    note = "TPIS78"
       )
```

```
species(name = "H",
    atoms = " H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.500000000E+00,   7.053328190E-
13,
               -1.995919640E-15,   2.300816320E-18,  -9.277323320E-22,
                2.547365990E+04,  -4.466828530E-01] ),
       NASA( [ 1000.00,  3500.00], [  2.500000010E+00,  -2.308429730E-
11,
                1.615619480E-14,  -4.735152350E-18,   4.981973570E-22,
                2.547365990E+04,  -4.466829140E-01] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     2.05,
                     well_depth =   145.00),
    note = "L 7/88"
       )

species(name = "O",
    atoms = " O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.168267100E+00,  -3.279318840E-
03,
                6.643063960E-06,  -6.128066240E-09,   2.112659710E-12,
                2.912225920E+04,   2.051933460E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.569420780E+00,  -8.597411370E-
05,
                4.194845890E-08,  -1.001777990E-11,   1.228336910E-15,
                2.921757910E+04,   4.784338640E+00] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "L 1/90"
       )

species(name = "O2",
    atoms = " O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.782456360E+00,  -2.996734160E-
03,
                9.847302010E-06,  -9.681295090E-09,   3.243728370E-12,
               -1.063943560E+03,   3.657675730E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.282537840E+00,   1.483087540E-
03,
               -7.579666690E-07,   2.094705550E-10,  -2.167177940E-14,
               -1.088457720E+03,   5.453231290E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.46,
                     well_depth =   107.40,
                     polar =     1.60,
                     rot_relax =     3.80),
```

```
    note = "TPIS89"
       )

species(name = "OH",
    atoms = " O:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.992015430E+00,  -2.401317520E-
03,
                4.617938410E-06,  -3.881133330E-09,   1.364114700E-12,
                3.615080560E+03,  -1.039254580E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.092887670E+00,   5.484297160E-
04,
                1.265052280E-07,  -8.794615560E-11,   1.174123760E-14,
                3.858657000E+03,   4.476696100E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "RUS 78"
       )

species(name = "H2O",
    atoms = " H:2  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.198640560E+00,  -2.036434100E-
03,
                6.520402110E-06,  -5.487970620E-09,   1.771978170E-12,
               -3.029372670E+04,  -8.490322080E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.033992490E+00,   2.176918040E-
03,
               -1.640725180E-07,  -9.704198700E-11,   1.682009920E-14,
               -3.000429710E+04,   4.966770100E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.60,
                     well_depth =   572.40,
                     dipole =     1.84,
                     rot_relax =     4.00),
    note = "L 8/89"
       )

species(name = "HO2",
    atoms = " H:1  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.301798010E+00,  -4.749120510E-
03,
                2.115828910E-05,  -2.427638940E-08,   9.292251240E-12,
                2.948080400E+02,   3.716662450E+00] ),
       NASA( [ 1000.00,  3500.00], [  4.017210900E+00,   2.239820130E-
03,
               -6.336581500E-07,   1.142463700E-10,  -1.079085350E-14,
                1.118567130E+02,   3.785102150E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
```

```
                    diam =     3.46,
                    well_depth =   107.40,
                    rot_relax =     1.00),
    note = "L 5/89"
       )

species(name = "H2O2",
    atoms = " H:2  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.276112690E+00,  -5.428224170E-
04,
                1.673357010E-05,  -2.157708130E-08,   8.624543630E-12,
               -1.770258210E+04,   3.435050740E+00] ),
       NASA( [ 1000.00,  3500.00], [  4.165002850E+00,   4.908316940E-
03,
               -1.901392250E-06,   3.711859860E-10,  -2.879083050E-14,
               -1.786178770E+04,   2.916156620E+00] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     3.46,
                    well_depth =   107.40,
                    rot_relax =     3.80),
    note = "L 7/88"
       )

species(name = "C",
    atoms = " C:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.554239550E+00,  -3.215377240E-
04,
                7.337922450E-07,  -7.322348890E-10,   2.665214460E-13,
                8.544388320E+04,   4.531308480E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.492668880E+00,   4.798892840E-
05,
               -7.243350200E-08,   3.742910290E-11,  -4.872778930E-15,
                8.545129530E+04,   4.801503730E+00] )
             ),
    transport = gas_transport(
                    geom = "atom",
                    diam =     3.30,
                    well_depth =    71.40),
    note = "L11/88"
       )

species(name = "CH",
    atoms = " C:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.489816650E+00,   3.238355410E-
04,
               -1.688990650E-06,   3.162173270E-09,  -1.406090670E-12,
                7.079729340E+04,   2.084011080E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.878464730E+00,   9.709136810E-
04,
                1.444456550E-07,  -1.306878490E-10,   1.760793830E-14,
                7.101243640E+04,   5.484979990E+00] )
             ),
```

```
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "TPIS79"
       )

species(name = "CH2",
    atoms = " C:1  H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.762678670E+00,   9.688721430E-
04,
                2.794898410E-06,  -3.850911530E-09,   1.687417190E-12,
                4.600404010E+04,   1.562531850E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.874101130E+00,   3.656392920E-
03,
               -1.408945970E-06,   2.601795490E-10,  -1.877275670E-14,
                4.626360400E+04,   6.171193240E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.80,
                     well_depth =   144.00),
    note = "L S/93"
       )

species(name = "CH2(S)",
    atoms = " C:1  H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.198604110E+00,  -2.366614190E-
03,
                8.232962200E-06,  -6.688159810E-09,   1.943147370E-12,
                5.049681630E+04,  -7.691189670E-01] ),
       NASA( [ 1000.00,  3500.00], [  2.292038420E+00,   4.655886370E-
03,
               -2.011919470E-06,   4.179060000E-10,  -3.397163650E-14,
                5.092599970E+04,   8.626501690E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.80,
                     well_depth =   144.00),
    note = "L S/93"
       )

species(name = "CH3",
    atoms = " C:1  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.673590400E+00,   2.010951750E-
03,
                5.730218560E-06,  -6.871174250E-09,   2.543857340E-12,
                1.644499880E+04,   1.604564330E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.285717720E+00,   7.239900370E-
03,
               -2.987143480E-06,   5.956846440E-10,  -4.671543940E-14,
                1.677558430E+04,   8.480071790E+00] )
             ),
```

```
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.80,
                     well_depth =   144.00),
    note = "L11/89"
       )

species(name = "CH4",
    atoms = " C:1  H:4 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  5.149876130E+00,  -1.367097880E-
02,
                4.918005990E-05,  -4.847430260E-08,   1.666939560E-11,
               -1.024664760E+04,  -4.641303760E+00] ),
       NASA( [ 1000.00,  3500.00], [  7.485149500E-02,   1.339094670E-
02,
               -5.732858090E-06,   1.222925350E-09,  -1.018152300E-13,
               -9.468344590E+03,   1.843731800E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.75,
                     well_depth =   141.40,
                     polar =     2.60,
                     rot_relax =    13.00),
    note = "L 8/88"
       )

species(name = "CO",
    atoms = " C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.579533470E+00,  -6.103536800E-
04,
                1.016814330E-06,   9.070058840E-10,  -9.044244990E-13,
               -1.434408600E+04,   3.508409280E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.715185610E+00,   2.062527430E-
03,
               -9.988257710E-07,   2.300530080E-10,  -2.036477160E-14,
               -1.415187240E+04,   7.818687720E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.65,
                     well_depth =    98.10,
                     polar =     1.95,
                     rot_relax =     1.80),
    note = "TPIS79"
       )

species(name = "CO2",
    atoms = " C:1  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.356773520E+00,   8.984596770E-
03,
               -7.123562690E-06,   2.459190220E-09,  -1.436995480E-13,
               -4.837196970E+04,   9.901052220E+00] ),
```

```
       NASA( [ 1000.00,  3500.00], [  3.857460290E+00,   4.414370260E-
03,
               -2.214814040E-06,   5.234901880E-10,  -4.720841640E-14,
               -4.875916600E+04,   2.271638060E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.76,
                     well_depth =   244.00,
                     polar =     2.65,
                     rot_relax =     2.10),
    note = "L 7/88"
       )

species(name = "HCO",
    atoms = " H:1  C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.221185840E+00,  -3.243925320E-
03,
                1.377994460E-05,  -1.331440930E-08,   4.337688650E-12,
                3.839564960E+03,   3.394372430E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.772174380E+00,   4.956955260E-
03,
               -2.484456130E-06,   5.891617780E-10,  -5.335087110E-14,
                4.011918150E+03,   9.798344920E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00),
    note = "L12/89"
       )

species(name = "CH2O",
    atoms = " H:2  C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.793723150E+00,  -9.908333690E-
03,
                3.732200080E-05,  -3.792852610E-08,   1.317726520E-11,
               -1.430895670E+04,   6.028129000E-01] ),
       NASA( [ 1000.00,  3500.00], [  1.760690080E+00,   9.200000820E-
03,
               -4.422588130E-06,   1.006412120E-09,  -8.838556400E-14,
               -1.399583230E+04,   1.365632300E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00,
                     rot_relax =     2.00),
    note = "L 8/88"
       )

species(name = "CH2OH",
    atoms = " C:1  H:3  O:1 ",
    thermo = (
```

```
       NASA( [  200.00,  1000.00], [  3.863889180E+00,   5.596723040E-
03,
                5.932717910E-06,  -1.045320120E-08,   4.369672780E-12,
               -3.193913670E+03,   5.473022430E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.692665690E+00,   8.645767970E-
03,
               -3.751011200E-06,   7.872346360E-10,  -6.485542010E-14,
               -3.242506270E+03,   5.810432150E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.69,
                     well_depth =   417.00,
                     dipole =     1.70,
                     rot_relax =     2.00),
    note = "GUNL93"
       )

species(name = "CH3O",
    atoms = " C:1  H:3  O:1 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  2.106204000E+00,   7.216595000E-
03,
                5.338472000E-06,  -7.377636000E-09,   2.075610000E-12,
                9.786011000E+02,   1.315217700E+01] ),
       NASA( [ 1000.00,  3000.00], [  3.770799000E+00,   7.871497000E-
03,
               -2.656384000E-06,   3.944431000E-10,  -2.112616000E-14,
                1.278325200E+02,   2.929575000E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.69,
                     well_depth =   417.00,
                     dipole =     1.70,
                     rot_relax =     2.00),
    note = "121686"
       )

species(name = "CH3OH",
    atoms = " C:1  H:4  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  5.715395820E+00,  -1.523091290E-
02,
                6.524411550E-05,  -7.108068890E-08,   2.613526980E-11,
               -2.564276560E+04,  -1.504098230E+00] ),
       NASA( [ 1000.00,  3500.00], [  1.789707910E+00,   1.409382920E-
02,
               -6.365008350E-06,   1.381710850E-09,  -1.170602200E-13,
               -2.537487470E+04,   1.450236230E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.63,
                     well_depth =   481.80,
                     rot_relax =     1.00),
    note = "L 8/88"
```

```
       )

species(name = "C2H",
    atoms = " C:2  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.889657330E+00,   1.340996110E-
02,
               -2.847695010E-05,   2.947910450E-08,  -1.093315110E-11,
                6.683939320E+04,   6.222964380E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.167806520E+00,   4.752219020E-
03,
               -1.837870770E-06,   3.041902520E-10,  -1.772327700E-14,
                6.712106500E+04,   6.635894750E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     4.10,
                     well_depth =   209.00,
                     rot_relax =     2.50),
    note = "L 1/91"
       )

species(name = "C2H2",
    atoms = " C:2  H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  8.086810940E-01,   2.336156290E-
02,
               -3.551718150E-05,   2.801524370E-08,  -8.500729740E-12,
                2.642898070E+04,   1.393970510E+01] ),
       NASA( [ 1000.00,  3500.00], [  4.147569640E+00,   5.961666640E-
03,
               -2.372948520E-06,   4.674121710E-10,  -3.612352130E-14,
                2.593599920E+04,  -1.230281210E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     4.10,
                     well_depth =   209.00,
                     rot_relax =     2.50),
    note = "L 1/91"
       )

species(name = "C2H3",
    atoms = " C:2  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.212466450E+00,   1.514791620E-
03,
                2.592094120E-05,  -3.576578470E-08,   1.471508730E-11,
                3.485984680E+04,   8.510540250E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.016724000E+00,   1.033022920E-
02,
               -4.680823490E-06,   1.017632880E-09,  -8.626070410E-14,
                3.461287390E+04,   7.787323780E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.10,
```

```
                    well_depth =   209.00,
                    rot_relax =     1.00),
    note = "L 2/92"
       )

species(name = "C2H4",
    atoms = " C:2  H:4 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.959201480E+00,  -7.570522470E-
03,
                5.709902920E-05,  -6.915887530E-08,   2.698843730E-11,
                5.089775930E+03,   4.097330960E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.036111160E+00,   1.464541510E-
02,
               -6.710779150E-06,   1.472229230E-09,  -1.257060610E-13,
                4.939886140E+03,   1.030536930E+01] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     3.97,
                    well_depth =   280.80,
                    rot_relax =     1.50),
    note = "L 1/91"
       )

species(name = "C2H5",
    atoms = " C:2  H:5 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.306465680E+00,  -4.186588920E-
03,
                4.971428070E-05,  -5.991266060E-08,   2.305090040E-11,
                1.284162650E+04,   4.707209240E+00] ),
       NASA( [ 1000.00,  3500.00], [  1.954656420E+00,   1.739727220E-
02,
               -7.982066680E-06,   1.752176890E-09,  -1.496415760E-13,
                1.285752000E+04,   1.346243430E+01] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     4.30,
                    well_depth =   252.30,
                    rot_relax =     1.50),
    note = "L12/92"
       )

species(name = "C2H6",
    atoms = " C:2  H:6 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.291424920E+00,  -5.501542700E-
03,
                5.994382880E-05,  -7.084662850E-08,   2.686857710E-11,
               -1.152220550E+04,   2.666823160E+00] ),
       NASA( [ 1000.00,  3500.00], [  1.071881500E+00,   2.168526770E-
02,
               -1.002560670E-05,   2.214120010E-09,  -1.900028900E-13,
               -1.142639320E+04,   1.511561070E+01] )
             ),
```

```
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.30,
                     well_depth =   252.30,
                     rot_relax =     1.50),
    note = "L 8/88"
       )

species(name = "HCCO",
    atoms = " H:1  C:2  O:1 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  2.251721400E+00,   1.765502100E-
02,
                -2.372910100E-05,   1.727575900E-08,  -5.066481100E-12,
                 2.005944900E+04,   1.249041700E+01] ),
       NASA( [ 1000.00,  4000.00], [  5.628205800E+00,   4.085340100E-
03,
                -1.593454700E-06,   2.862605200E-10,  -1.940783200E-14,
                 1.932721500E+04,  -3.930259500E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.50,
                     well_depth =   150.00,
                     rot_relax =     1.00),
    note = "SRIC91"
       )

species(name = "CH2CO",
    atoms = " C:2  H:2  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.135836300E+00,   1.811887210E-
02,
                -1.739474740E-05,   9.343975680E-09,  -2.014576150E-12,
                -7.042918040E+03,   1.221564800E+01] ),
       NASA( [ 1000.00,  3500.00], [  4.511297320E+00,   9.003597450E-
03,
                -4.169396350E-06,   9.233458820E-10,  -7.948382010E-14,
                -7.551053110E+03,   6.322472050E-01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.97,
                     well_depth =   436.00,
                     rot_relax =     2.00),
    note = "L 5/90"
       )

species(name = "HCCOH",
    atoms = " C:2  O:1  H:2 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  1.242373300E+00,   3.107220100E-
02,
                -5.086686400E-05,   4.313713100E-08,  -1.401459400E-11,
                 8.031614300E+03,   1.387431900E+01] ),
       NASA( [ 1000.00,  5000.00], [  5.923829100E+00,   6.792360000E-
03,
```

```
               -2.565856400E-06,   4.498784100E-10,  -2.994010100E-14,
                7.264626000E+03,  -7.601774200E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.97,
                     well_depth =   436.00,
                     rot_relax =     2.00),
    note = "SRI91"
       )

species(name = "N",
    atoms = " N:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
                5.610463700E+04,   4.193908700E+00] ),
       NASA( [ 1000.00,  6000.00], [  2.415942900E+00,   1.748906500E-
04,
               -1.190236900E-07,   3.022624500E-11,  -2.036098200E-15,
                5.613377300E+04,   4.649609600E+00] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     3.30,
                     well_depth =    71.40),
    note = "L 6/88"
       )

species(name = "NH",
    atoms = " N:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.492908500E+00,   3.117919800E-
04,
               -1.489048400E-06,   2.481644200E-09,  -1.035696700E-12,
                4.188062900E+04,   1.848327800E+00] ),
       NASA( [ 1000.00,  6000.00], [  2.783692800E+00,   1.329843000E-
03,
               -4.247804700E-07,   7.834850100E-11,  -5.504447000E-15,
                4.212084800E+04,   5.740779900E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.65,
                     well_depth =    80.00,
                     rot_relax =     4.00),
    note = "And94"
       )

species(name = "NH2",
    atoms = " N:1  H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.204002900E+00,  -2.106138500E-
03,
               7.106834800E-06,  -5.611519700E-09,   1.644071700E-12,
               2.188591000E+04,  -1.418424800E-01] ),
```

```
       NASA( [ 1000.00,  6000.00], [  2.834742100E+00,   3.207308200E-
03,
                -9.339080400E-07,   1.370295300E-10,  -7.920614400E-15,
                 2.217195700E+04,   6.520416300E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.65,
                     well_depth =    80.00,
                     polar =     2.26,
                     rot_relax =     4.00),
    note = "And89"
       )

species(name = "NH3",
    atoms = " N:1  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.286027400E+00,  -4.660523000E-
03,
                 2.171851300E-05,  -2.280888700E-08,   8.263804600E-12,
                -6.741728500E+03,  -6.253727700E-01] ),
       NASA( [ 1000.00,  6000.00], [  2.634452100E+00,   5.666256000E-
03,
                -1.727867600E-06,   2.386716100E-10,  -1.257878600E-14,
                -6.544695800E+03,   6.566292800E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.92,
                     well_depth =   481.00,
                     dipole =     1.47,
                     rot_relax =    10.00),
    note = "J 6/77"
       )

species(name = "NNH",
    atoms = " N:2  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.344692700E+00,  -4.849707200E-
03,
                 2.005945900E-05,  -2.172646400E-08,   7.946953900E-12,
                 2.879197300E+04,   2.977941000E+00] ),
       NASA( [ 1000.00,  6000.00], [  3.766754400E+00,   2.891508200E-
03,
                -1.041662000E-06,   1.684259400E-10,  -1.009189600E-14,
                 2.865069700E+04,   4.470506700E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.80,
                     well_depth =    71.40,
                     rot_relax =     1.00),
    note = "T07/93"
       )

species(name = "NO",
    atoms = " N:1  O:1 ",
```

```
    thermo = (
       NASA( [  200.00,  1000.00], [  4.218476300E+00,  -4.638976000E-
03,
                1.104102200E-05,  -9.336135400E-09,   2.803577000E-12,
                9.844623000E+03,   2.280846400E+00] ),
       NASA( [ 1000.00,  6000.00], [  3.260605600E+00,   1.191104300E-
03,
               -4.291704800E-07,   6.945766900E-11,  -4.033609900E-15,
                9.920974600E+03,   6.369302700E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.62,
                     well_depth =    97.53,
                     polar =     1.76,
                     rot_relax =     4.00),
    note = "RUS 78"
       )

species(name = "NO2",
    atoms = " N:1  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.944031200E+00,  -1.585429000E-
03,
                1.665781200E-05,  -2.047542600E-08,   7.835056400E-12,
                2.896617900E+03,   6.311991700E+00] ),
       NASA( [ 1000.00,  6000.00], [  4.884754200E+00,   2.172395600E-
03,
               -8.280690600E-07,   1.574751000E-10,  -1.051089500E-14,
                2.316498300E+03,  -1.174169500E-01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.50,
                     well_depth =   200.00,
                     rot_relax =     1.00),
    note = "L 7/88"
       )

species(name = "N2O",
    atoms = " N:2  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.257150200E+00,   1.130472800E-
02,
               -1.367131900E-05,   9.681980600E-09,  -2.930718200E-12,
                8.741774400E+03,   1.075799200E+01] ),
       NASA( [ 1000.00,  6000.00], [  4.823072900E+00,   2.627025100E-
03,
               -9.585087400E-07,   1.600071200E-10,  -9.775230300E-15,
                8.073404800E+03,  -2.201720700E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.83,
                     well_depth =   232.40,
                     rot_relax =     1.00),
    note = "L 7/88"
```

```
       )

species(name = "HNO",
    atoms = " H:1  N:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.533491600E+00,  -5.669617100E-
03,
                1.847320700E-05,  -1.713709400E-08,   5.545457300E-12,
                1.154829700E+04,   1.749841700E+00] ),
       NASA( [ 1000.00,  6000.00], [  2.979250900E+00,   3.494405900E-
03,
               -7.854977800E-07,   5.747959400E-11,  -1.933591600E-16,
                1.175058200E+04,   8.606372800E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.49,
                     well_depth =   116.70,
                     rot_relax =     1.00),
    note = "And93"
       )

species(name = "CN",
    atoms = " C:1  N:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.612935100E+00,  -9.555132700E-
04,
                2.144297700E-06,  -3.151632300E-10,  -4.643035600E-13,
                5.170834000E+04,   3.980499500E+00] ),
       NASA( [ 1000.00,  6000.00], [  3.745980500E+00,   4.345077500E-
05,
                2.970598400E-07,  -6.865180600E-11,   4.413417300E-15,
                5.153618800E+04,   2.786760100E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.86,
                     well_depth =    75.00,
                     rot_relax =     1.00),
    note = "HBH92"
       )

species(name = "HCN",
    atoms = " H:1  C:1  N:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.258988600E+00,   1.005117000E-
02,
               -1.335176300E-05,   1.009234900E-08,  -3.008902800E-12,
                1.471263300E+04,   8.916441900E+00] ),
       NASA( [ 1000.00,  6000.00], [  3.802239200E+00,   3.146422800E-
03,
               -1.063218500E-06,   1.661975700E-10,  -9.799757000E-15,
                1.440729200E+04,   1.575460100E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.63,
```

```
                    well_depth =   569.00,
                    rot_relax =     1.00),
    note = "GRI/98"
       )

species(name = "H2CN",
    atoms = " H:2  C:1  N:1 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  2.851661000E+00,   5.695233100E-
03,
                1.071140000E-06,  -1.622612000E-09,  -2.351108100E-13,
                2.863782000E+04,   8.992751100E+00] ),
       NASA( [ 1000.00,  4000.00], [  5.209703000E+00,   2.969291100E-
03,
               -2.855589100E-07,  -1.635550000E-10,   3.043258900E-14,
                2.767710900E+04,  -4.444478000E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.63,
                    well_depth =   569.00,
                    rot_relax =     1.00),
    note = "41687"
       )

species(name = "HCNN",
    atoms = " C:1  N:2  H:1 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  2.524319400E+00,   1.596061900E-
02,
               -1.881635400E-05,   1.212554000E-08,  -3.235737800E-12,
                5.426198400E+04,   1.167587000E+01] ),
       NASA( [ 1000.00,  5000.00], [  5.894636200E+00,   3.989595900E-
03,
               -1.598238000E-06,   2.924939500E-10,  -2.009468600E-14,
                5.345294100E+04,  -5.103050200E+00] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     2.50,
                    well_depth =   150.00,
                    rot_relax =     1.00),
    note = "SRI/94"
       )

species(name = "HCNO",
    atoms = " H:1  N:1  C:1  O:1 ",
    thermo = (
       NASA( [  300.00,  1382.00], [  2.647279890E+00,   1.275053420E-
02,
               -1.047942360E-05,   4.414328360E-09,  -7.575214660E-13,
                1.929902520E+04,   1.073329720E+01] ),
       NASA( [ 1382.00,  5000.00], [  6.598604560E+00,   3.027786260E-
03,
               -1.077043460E-06,   1.716665280E-10,  -1.014393910E-14,
                1.796613390E+04,  -1.033065990E+01] )
             ),
```

```
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.83,
                     well_depth =   232.40,
                     rot_relax =     1.00),
    note = "BDEA94"
       )

species(name = "HOCN",
    atoms = " H:1  N:1  C:1  O:1 ",
    thermo = (
       NASA( [  300.00,  1368.00], [  3.786049520E+00,   6.886679220E-
03,
                -3.214878640E-06,   5.171957670E-10,   1.193607880E-14,
                -2.826984000E+03,   5.632921620E+00] ),
       NASA( [ 1368.00,  5000.00], [  5.897848850E+00,   3.167893930E-
03,
                -1.118010640E-06,   1.772431440E-10,  -1.043391770E-14,
                -3.706533310E+03,  -6.181678250E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.83,
                     well_depth =   232.40,
                     rot_relax =     1.00),
    note = "BDEA94"
       )

species(name = "HNCO",
    atoms = " H:1  N:1  C:1  O:1 ",
    thermo = (
       NASA( [  300.00,  1478.00], [  3.630963170E+00,   7.302823570E-
03,
                -2.280500030E-06,  -6.612712980E-10,   3.622357520E-13,
                -1.558736360E+04,   6.194577270E+00] ),
       NASA( [ 1478.00,  5000.00], [  6.223951340E+00,   3.178640040E-
03,
                -1.093787550E-06,   1.707351630E-10,  -9.950219550E-15,
                -1.665993440E+04,  -8.382247410E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.83,
                     well_depth =   232.40,
                     rot_relax =     1.00),
    note = "BDEA94"
       )

species(name = "NCO",
    atoms = " N:1  C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.826930800E+00,   8.805168800E-
03,
                -8.386613400E-06,   4.801696400E-09,  -1.331359500E-12,
                 1.468247700E+04,   9.550464600E+00] ),
       NASA( [ 1000.00,  6000.00], [  5.152184500E+00,   2.305176100E-
03,
```

```
               -8.803315300E-07,   1.478909800E-10,  -9.097799600E-15,
                1.400412300E+04,  -2.544266000E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.83,
                     well_depth =   232.40,
                     rot_relax =     1.00),
    note = "EA 93"
       )

species(name = "N2",
    atoms = " N:2 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  3.298677000E+00,   1.408240400E-
03,
               -3.963222000E-06,   5.641515000E-09,  -2.444854000E-12,
               -1.020899900E+03,   3.950372000E+00] ),
       NASA( [ 1000.00,  5000.00], [  2.926640000E+00,   1.487976800E-
03,
               -5.684760000E-07,   1.009703800E-10,  -6.753351000E-15,
               -9.227977000E+02,   5.980528000E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.62,
                     well_depth =    97.53,
                     polar =     1.76,
                     rot_relax =     4.00),
    note = "121286"
       )

species(name = "AR",
    atoms = " Ar:1 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
               -7.453750000E+02,   4.366000000E+00] ),
       NASA( [ 1000.00,  5000.00], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
               -7.453750000E+02,   4.366000000E+00] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     3.33,
                     well_depth =   136.50),
    note = "120186"
       )

species(name = "C3H7",
    atoms = " C:3  H:7 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  1.051551800E+00,   2.599198000E-
02,
                2.380054000E-06,  -1.960956900E-08,   9.373247000E-12,
```

```
                1.063186300E+04,   2.112255900E+01] ),
       NASA( [ 1000.00,  5000.00], [  7.702698700E+00,   1.604420300E-
02,
               -5.283322000E-06,   7.629859000E-10,  -3.939228400E-14,
                8.298433600E+03,  -1.548018000E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.98,
                     well_depth =   266.80,
                     rot_relax =     1.00),
    note = "L 9/84"
       )

species(name = "C3H8",
    atoms = " C:3  H:8 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  9.335538100E-01,   2.642457900E-
02,
                6.105972700E-06,  -2.197749900E-08,   9.514925300E-12,
               -1.395852000E+04,   1.920169100E+01] ),
       NASA( [ 1000.00,  5000.00], [  7.534136800E+00,   1.887223900E-
02,
               -6.271849100E-06,   9.147564900E-10,  -4.783806900E-14,
               -1.646751600E+04,  -1.789234900E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.98,
                     well_depth =   266.80,
                     rot_relax =     1.00),
    note = "L 4/85"
       )

species(name = "CH2CHO",
    atoms = " O:1  H:3  C:2 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  3.409062000E+00,   1.073857400E-
02,
                1.891492000E-06,  -7.158583000E-09,   2.867385000E-12,
                1.521476600E+03,   9.558290000E+00] ),
       NASA( [ 1000.00,  5000.00], [  5.975670000E+00,   8.130591000E-
03,
               -2.743624000E-06,   4.070304000E-10,  -2.176017000E-14,
                4.903218000E+02,  -5.045251000E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.97,
                     well_depth =   436.00,
                     rot_relax =     2.00),
    note = "SAND86"
       )

species(name = "CH3CHO",
    atoms = " C:2  H:4  O:1 ",
    thermo = (
```

```
       NASA( [  200.00,  1000.00], [  4.729459500E+00,  -3.193285800E-
03,
                4.753492100E-05,  -5.745861100E-08,   2.193111200E-11,
               -2.157287800E+04,   4.103015900E+00] ),
       NASA( [ 1000.00,  6000.00], [  5.404110800E+00,   1.172305900E-
02,
               -4.226313700E-06,   6.837245100E-10,  -4.098486300E-14,
               -2.259312200E+04,  -3.480791700E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.97,
                     well_depth =   436.00,
                     rot_relax =     2.00),
    note = "L 8/88"
       )

species(name = "E",
    atoms = " E:1 ",
    thermo = (
       NASA( [  200.00,     5.49], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
               -7.453750000E+02,  -1.172469020E+01] ),
       NASA( [    5.49,  6000.00], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
               -7.453750000E+02,  -1.172469020E+01] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =   425.00,
                     well_depth =   850.00,
                     rot_relax =     1.00),
    note = "L10/92"
       )

species(name = "HCO+",
    atoms = " H:1  C:1  O:1  E:-1 ",
    thermo = (
       NASA( [  300.00,    29.02], [  2.473973600E+00,   8.671559000E-
03,
               -1.003150000E-05,   6.717052700E-09,  -1.787267400E-12,
                9.914660800E+04,   8.175711870E+00] ),
       NASA( [   29.02,  5000.00], [  3.741188000E+00,   3.344151700E-
03,
               -1.239712100E-06,   2.118938800E-10,  -1.370415000E-14,
                9.888407800E+04,   2.078613570E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00),
    note = "J12/70"
       )

species(name = "H3O+",
```

```
    atoms = " H:3  O:1  E:-1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.198640560E+00,  -2.036434100E-
03,
                6.520402110E-06,  -5.487970620E-09,   1.771978170E-12,
               -3.029372670E+04,  -8.490322080E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.033992490E+00,   2.176918040E-
03,
               -1.640725180E-07,  -9.704198700E-11,   1.682009920E-14,
               -3.000429710E+04,   4.966770100E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.60,
                     well_depth =   572.40,
                     dipole =     1.84,
                     rot_relax =     4.00),
    note = "L 8/89"
       )

#-----------------------------------------------------------------------
---------
#  Reaction data
#-----------------------------------------------------------------------
---------

#  Reaction 1
three_body_reaction( "2 O + M <=> O2 + M",  [1.20000E+17, -1, 0],
         efficiencies = " AR:0.83  C2H6:3  CH4:2  CO:1.75  CO2:3.6
H2:2.4  H2O:15.4 ")

#  Reaction 2
three_body_reaction( "O + H + M <=> OH + M",  [5.00000E+17, -1, 0],
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 3
reaction(  "O + H2 <=> H + OH",  [3.87000E+04, 2.7, 6260])

#  Reaction 4
reaction(  "O + HO2 <=> OH + O2",  [2.00000E+13, 0, 0])

#  Reaction 5
reaction(  "O + H2O2 <=> OH + HO2",  [9.63000E+06, 2, 4000])

#  Reaction 6
reaction(  "O + CH <=> H + CO",  [5.70000E+13, 0, 0])

#  Reaction 7
reaction(  "O + CH2 <=> H + HCO",  [8.00000E+13, 0, 0])

#  Reaction 8
reaction(  "O + CH2(S) <=> H2 + CO",  [1.50000E+13, 0, 0])

#  Reaction 9
reaction(  "O + CH2(S) <=> H + HCO",  [1.50000E+13, 0, 0])
```

```
#  Reaction 10
reaction(  "O + CH3 <=> H + CH2O",  [5.06000E+13, 0, 0])

#  Reaction 11
reaction(  "O + CH4 <=> OH + CH3",  [1.02000E+09, 1.5, 8600])

#  Reaction 12
falloff_reaction( "O + CO (+ M) <=> CO2 (+ M)",
         kf = [1.80000E+10, 0, 2385],
         kf0   = [6.02000E+14, 0, 3000],
         efficiencies = " AR:0.5  C2H6:3  CH4:2  CO:1.5  CO2:3.5  H2:2
H2O:6  O2:6 ")

#  Reaction 13
reaction(  "O + HCO <=> OH + CO",  [3.00000E+13, 0, 0])

#  Reaction 14
reaction(  "O + HCO <=> H + CO2",  [3.00000E+13, 0, 0])

#  Reaction 15
reaction(  "O + CH2O <=> OH + HCO",  [3.90000E+13, 0, 3540])

#  Reaction 16
reaction(  "O + CH2OH <=> OH + CH2O",  [1.00000E+13, 0, 0])

#  Reaction 17
reaction(  "O + CH3O <=> OH + CH2O",  [1.00000E+13, 0, 0])

#  Reaction 18
reaction(  "O + CH3OH <=> OH + CH2OH",  [3.88000E+05, 2.5, 3100])

#  Reaction 19
reaction(  "O + CH3OH <=> OH + CH3O",  [1.30000E+05, 2.5, 5000])

#  Reaction 20
reaction(  "O + C2H <=> CH + CO",  [5.00000E+13, 0, 0])

#  Reaction 21
reaction(  "O + C2H2 <=> H + HCCO",  [1.35000E+07, 2, 1900])

#  Reaction 22
reaction(  "O + C2H2 <=> OH + C2H",  [4.60000E+19, -1.41, 28950])

#  Reaction 23
reaction(  "O + C2H2 <=> CO + CH2",  [6.94000E+06, 2, 1900])

#  Reaction 24
reaction(  "O + C2H3 <=> H + CH2CO",  [3.00000E+13, 0, 0])

#  Reaction 25
reaction(  "O + C2H4 <=> CH3 + HCO",  [1.25000E+07, 1.83, 220])

#  Reaction 26
reaction(  "O + C2H5 <=> CH3 + CH2O",  [2.24000E+13, 0, 0])

#  Reaction 27
reaction(  "O + C2H6 <=> OH + C2H5",  [8.98000E+07, 1.92, 5690])
```

```
#  Reaction 28
reaction(  "O + HCCO <=> H + 2 CO",  [1.00000E+14, 0, 0])

#  Reaction 29
reaction(  "O + CH2CO <=> OH + HCCO",  [1.00000E+13, 0, 8000])

#  Reaction 30
reaction(  "O + CH2CO <=> CH2 + CO2",  [1.75000E+12, 0, 1350])

#  Reaction 31
reaction(  "O2 + CO <=> O + CO2",  [2.50000E+12, 0, 47800])

#  Reaction 32
reaction(  "O2 + CH2O <=> HO2 + HCO",  [1.00000E+14, 0, 40000])

#  Reaction 33
three_body_reaction( "H + O2 + M <=> HO2 + M",  [2.80000E+18, -0.86,
0],
         efficiencies = " AR:0  C2H6:1.5  CO:0.75  CO2:1.5  H2O:0  N2:0
O2:0 ")

#  Reaction 34
reaction(  "H + 2 O2 <=> HO2 + O2",  [2.08000E+19, -1.24, 0])

#  Reaction 35
reaction(  "H + O2 + H2O <=> HO2 + H2O",  [1.12600E+19, -0.76, 0])

#  Reaction 36
reaction(  "H + O2 + N2 <=> HO2 + N2",  [2.60000E+19, -1.24, 0])

#  Reaction 37
reaction(  "H + O2 + AR <=> HO2 + AR",  [7.00000E+17, -0.8, 0])

#  Reaction 38
reaction(  "H + O2 <=> O + OH",  [2.65000E+16, -0.6707, 17041])

#  Reaction 39
three_body_reaction( "2 H + M <=> H2 + M",  [1.00000E+18, -1, 0],
         efficiencies = " AR:0.63  C2H6:3  CH4:2  CO2:0  H2:0  H2O:0 ")

#  Reaction 40
reaction(  "2 H + H2 <=> 2 H2",  [9.00000E+16, -0.6, 0])

#  Reaction 41
reaction(  "2 H + H2O <=> H2 + H2O",  [6.00000E+19, -1.25, 0])

#  Reaction 42
reaction(  "2 H + CO2 <=> H2 + CO2",  [5.50000E+20, -2, 0])

#  Reaction 43
three_body_reaction( "H + OH + M <=> H2O + M",  [2.20000E+22, -2, 0],
         efficiencies = " AR:0.38  C2H6:3  CH4:2  H2:0.73  H2O:3.65 ")

#  Reaction 44
reaction(  "H + HO2 <=> O + H2O",  [3.97000E+12, 0, 671])
```

```
#  Reaction 45
reaction(  "H + HO2 <=> O2 + H2",  [4.48000E+13, 0, 1068])

#  Reaction 46
reaction(  "H + HO2 <=> 2 OH",  [8.40000E+13, 0, 635])

#  Reaction 47
reaction(  "H + H2O2 <=> HO2 + H2",  [1.21000E+07, 2, 5200])

#  Reaction 48
reaction(  "H + H2O2 <=> OH + H2O",  [1.00000E+13, 0, 3600])

#  Reaction 49
reaction(  "H + CH <=> C + H2",  [1.65000E+14, 0, 0])

#  Reaction 50
falloff_reaction( "H + CH2 (+ M) <=> CH3 (+ M)",
         kf = [6.00000E+14, 0, 0],
         kf0   = [1.04000E+26, -2.76, 1600],
         falloff = Troe(A = 0.562, T3 = 91, T1 = 5836, T2 = 8552),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 51
reaction(  "H + CH2(S) <=> CH + H2",  [3.00000E+13, 0, 0])

#  Reaction 52
falloff_reaction( "H + CH3 (+ M) <=> CH4 (+ M)",
         kf = [1.39000E+16, -0.534, 536],
         kf0   = [2.62000E+33, -4.76, 2440],
         falloff = Troe(A = 0.783, T3 = 74, T1 = 2941, T2 = 6964),
         efficiencies = " AR:0.7  C2H6:3  CH4:3  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 53
reaction(  "H + CH4 <=> CH3 + H2",  [6.60000E+08, 1.62, 10840])

#  Reaction 54
falloff_reaction( "H + HCO (+ M) <=> CH2O (+ M)",
         kf = [1.09000E+12, 0.48, -260],
         kf0   = [2.47000E+24, -2.57, 425],
         falloff = Troe(A = 0.7824, T3 = 271, T1 = 2755, T2 = 6570),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 55
reaction(  "H + HCO <=> H2 + CO",  [7.34000E+13, 0, 0])

#  Reaction 56
falloff_reaction( "H + CH2O (+ M) <=> CH2OH (+ M)",
         kf = [5.40000E+11, 0.454, 3600],
         kf0   = [1.27000E+32, -4.82, 6530],
         falloff = Troe(A = 0.7187, T3 = 103, T1 = 1291, T2 = 4160),
         efficiencies = " C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2  H2O:6 ")

#  Reaction 57
falloff_reaction( "H + CH2O (+ M) <=> CH3O (+ M)",
```

```
         kf = [5.40000E+11, 0.454, 2600],
         kf0   = [2.20000E+30, -4.8, 5560],
         falloff = Troe(A = 0.758, T3 = 94, T1 = 1555, T2 = 4200),
         efficiencies = " C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2  H2O:6 ")

#  Reaction 58
reaction(  "H + CH2O <=> HCO + H2",  [5.74000E+07, 1.9, 2742])

#  Reaction 59
falloff_reaction( "H + CH2OH (+ M) <=> CH3OH (+ M)",
         kf = [1.05500E+12, 0.5, 86],
         kf0   = [4.36000E+31, -4.65, 5080],
         falloff = Troe(A = 0.6, T3 = 100, T1 = 90000, T2 = 10000),
         efficiencies = " C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2  H2O:6 ")

#  Reaction 60
reaction(  "H + CH2OH <=> H2 + CH2O",  [2.00000E+13, 0, 0])

#  Reaction 61
reaction(  "H + CH2OH <=> OH + CH3",  [1.65000E+11, 0.65, -284])

#  Reaction 62
reaction(  "H + CH2OH <=> CH2(S) + H2O",  [3.28000E+13, -0.09, 610])

#  Reaction 63
falloff_reaction( "H + CH3O (+ M) <=> CH3OH (+ M)",
         kf = [2.43000E+12, 0.515, 50],
         kf0   = [4.66000E+41, -7.44, 14080],
         falloff = Troe(A = 0.7, T3 = 100, T1 = 90000, T2 = 10000),
         efficiencies = " C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2  H2O:6 ")

#  Reaction 64
reaction(  "H + CH3O <=> H + CH2OH",  [4.15000E+07, 1.63, 1924])

#  Reaction 65
reaction(  "H + CH3O <=> H2 + CH2O",  [2.00000E+13, 0, 0])

#  Reaction 66
reaction(  "H + CH3O <=> OH + CH3",  [1.50000E+12, 0.5, -110])

#  Reaction 67
reaction(  "H + CH3O <=> CH2(S) + H2O",  [2.62000E+14, -0.23, 1070])

#  Reaction 68
reaction(  "H + CH3OH <=> CH2OH + H2",  [1.70000E+07, 2.1, 4870])

#  Reaction 69
reaction(  "H + CH3OH <=> CH3O + H2",  [4.20000E+06, 2.1, 4870])

#  Reaction 70
falloff_reaction( "H + C2H (+ M) <=> C2H2 (+ M)",
         kf = [1.00000E+17, -1, 0],
         kf0   = [3.75000E+33, -4.8, 1900],
         falloff = Troe(A = 0.6464, T3 = 132, T1 = 1315, T2 = 5566),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")
```

```
#  Reaction 71
falloff_reaction( "H + C2H2 (+ M) <=> C2H3 (+ M)",
         kf = [5.60000E+12, 0, 2400],
         kf0   = [3.80000E+40, -7.27, 7220],
         falloff = Troe(A = 0.7507, T3 = 98.5, T1 = 1302, T2 = 4167),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 72
falloff_reaction( "H + C2H3 (+ M) <=> C2H4 (+ M)",
         kf = [6.08000E+12, 0.27, 280],
         kf0   = [1.40000E+30, -3.86, 3320],
         falloff = Troe(A = 0.782, T3 = 207.5, T1 = 2663, T2 = 6095),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 73
reaction(  "H + C2H3 <=> H2 + C2H2",  [3.00000E+13, 0, 0])

#  Reaction 74
falloff_reaction( "H + C2H4 (+ M) <=> C2H5 (+ M)",
         kf = [5.40000E+11, 0.454, 1820],
         kf0   = [6.00000E+41, -7.62, 6970],
         falloff = Troe(A = 0.9753, T3 = 210, T1 = 984, T2 = 4374),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 75
reaction(  "H + C2H4 <=> C2H3 + H2",  [1.32500E+06, 2.53, 12240])

#  Reaction 76
falloff_reaction( "H + C2H5 (+ M) <=> C2H6 (+ M)",
         kf = [5.21000E+17, -0.99, 1580],
         kf0   = [1.99000E+41, -7.08, 6685],
         falloff = Troe(A = 0.8422, T3 = 125, T1 = 2219, T2 = 6882),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 77
reaction(  "H + C2H5 <=> H2 + C2H4",  [2.00000E+12, 0, 0])

#  Reaction 78
reaction(  "H + C2H6 <=> C2H5 + H2",  [1.15000E+08, 1.9, 7530])

#  Reaction 79
reaction(  "H + HCCO <=> CH2(S) + CO",  [1.00000E+14, 0, 0])

#  Reaction 80
reaction(  "H + CH2CO <=> HCCO + H2",  [5.00000E+13, 0, 8000])

#  Reaction 81
reaction(  "H + CH2CO <=> CH3 + CO",  [1.13000E+13, 0, 3428])

#  Reaction 82
reaction(  "H + HCCOH <=> H + CH2CO",  [1.00000E+13, 0, 0])

#  Reaction 83
```

```
falloff_reaction( "H2 + CO (+ M) <=> CH2O (+ M)",
         kf = [4.30000E+07, 1.5, 79600],
         kf0   = [5.07000E+27, -3.42, 84350],
         falloff = Troe(A = 0.932, T3 = 197, T1 = 1540, T2 = 10300),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 84
reaction(  "OH + H2 <=> H + H2O",  [2.16000E+08, 1.51, 3430])

#  Reaction 85
falloff_reaction( "2 OH (+ M) <=> H2O2 (+ M)",
         kf = [7.40000E+13, -0.37, 0],
         kf0   = [2.30000E+18, -0.9, -1700],
         falloff = Troe(A = 0.7346, T3 = 94, T1 = 1756, T2 = 5182),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 86
reaction(  "2 OH <=> O + H2O",  [3.57000E+04, 2.4, -2110])

#  Reaction 87
reaction(  "OH + HO2 <=> O2 + H2O",  [1.45000E+13, 0, -500],
         options = ["duplicate"])

#  Reaction 88
reaction(  "OH + H2O2 <=> HO2 + H2O",  [2.00000E+12, 0, 427],
         options = ["duplicate"])

#  Reaction 89
reaction(  "OH + H2O2 <=> HO2 + H2O",  [1.70000E+18, 0, 29410],
         options = ["duplicate"])

#  Reaction 90
reaction(  "OH + C <=> H + CO",  [5.00000E+13, 0, 0])

#  Reaction 91
reaction(  "OH + CH <=> H + HCO",  [3.00000E+13, 0, 0])

#  Reaction 92
reaction(  "OH + CH2 <=> H + CH2O",  [2.00000E+13, 0, 0])

#  Reaction 93
reaction(  "OH + CH2 <=> CH + H2O",  [1.13000E+07, 2, 3000])

#  Reaction 94
reaction(  "OH + CH2(S) <=> H + CH2O",  [3.00000E+13, 0, 0])

#  Reaction 95
falloff_reaction( "OH + CH3 (+ M) <=> CH3OH (+ M)",
         kf = [2.79000E+18, -1.43, 1330],
         kf0   = [4.00000E+36, -5.92, 3140],
         falloff = Troe(A = 0.412, T3 = 195, T1 = 5900, T2 = 6394),
         efficiencies = " C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2  H2O:6 ")

#  Reaction 96
reaction(  "OH + CH3 <=> CH2 + H2O",  [5.60000E+07, 1.6, 5420])
```

```
#  Reaction 97
reaction(  "OH + CH3 <=> CH2(S) + H2O",  [6.44000E+17, -1.34, 1417])

#  Reaction 98
reaction(  "OH + CH4 <=> CH3 + H2O",  [1.00000E+08, 1.6, 3120])

#  Reaction 99
reaction(  "OH + CO <=> H + CO2",  [4.76000E+07, 1.228, 70])

#  Reaction 100
reaction(  "OH + HCO <=> H2O + CO",  [5.00000E+13, 0, 0])

#  Reaction 101
reaction(  "OH + CH2O <=> HCO + H2O",  [3.43000E+09, 1.18, -447])

#  Reaction 102
reaction(  "OH + CH2OH <=> H2O + CH2O",  [5.00000E+12, 0, 0])

#  Reaction 103
reaction(  "OH + CH3O <=> H2O + CH2O",  [5.00000E+12, 0, 0])

#  Reaction 104
reaction(  "OH + CH3OH <=> CH2OH + H2O",  [1.44000E+06, 2, -840])

#  Reaction 105
reaction(  "OH + CH3OH <=> CH3O + H2O",  [6.30000E+06, 2, 1500])

#  Reaction 106
reaction(  "OH + C2H <=> H + HCCO",  [2.00000E+13, 0, 0])

#  Reaction 107
reaction(  "OH + C2H2 <=> H + CH2CO",  [2.18000E-04, 4.5, -1000])

#  Reaction 108
reaction(  "OH + C2H2 <=> H + HCCOH",  [5.04000E+05, 2.3, 13500])

#  Reaction 109
reaction(  "OH + C2H2 <=> C2H + H2O",  [3.37000E+07, 2, 14000])

#  Reaction 110
reaction(  "OH + C2H2 <=> CH3 + CO",  [4.83000E-04, 4, -2000])

#  Reaction 111
reaction(  "OH + C2H3 <=> H2O + C2H2",  [5.00000E+12, 0, 0])

#  Reaction 112
reaction(  "OH + C2H4 <=> C2H3 + H2O",  [3.60000E+06, 2, 2500])

#  Reaction 113
reaction(  "OH + C2H6 <=> C2H5 + H2O",  [3.54000E+06, 2.12, 870])

#  Reaction 114
reaction(  "OH + CH2CO <=> HCCO + H2O",  [7.50000E+12, 0, 2000])

#  Reaction 115
reaction(  "2 HO2 <=> O2 + H2O2",  [1.30000E+11, 0, -1630],
```

```
         options = ["duplicate"])

#  Reaction 116
reaction(  "2 HO2 <=> O2 + H2O2",  [4.20000E+14, 0, 12000],
         options = ["duplicate"])

#  Reaction 117
reaction(  "HO2 + CH2 <=> OH + CH2O",  [2.00000E+13, 0, 0])

#  Reaction 118
reaction(  "HO2 + CH3 <=> O2 + CH4",  [1.00000E+12, 0, 0])

#  Reaction 119
reaction(  "HO2 + CH3 <=> OH + CH3O",  [3.78000E+13, 0, 0])

#  Reaction 120
reaction(  "HO2 + CO <=> OH + CO2",  [1.50000E+14, 0, 23600])

#  Reaction 121
reaction(  "HO2 + CH2O <=> HCO + H2O2",  [5.60000E+06, 2, 12000])

#  Reaction 122
reaction(  "C + O2 <=> O + CO",  [5.80000E+13, 0, 576])

#  Reaction 123
reaction(  "C + CH2 <=> H + C2H",  [5.00000E+13, 0, 0])

#  Reaction 124
reaction(  "C + CH3 <=> H + C2H2",  [5.00000E+13, 0, 0])

#  Reaction 125
reaction(  "CH + O2 <=> O + HCO",  [6.71000E+13, 0, 0])

#  Reaction 126
reaction(  "CH + H2 <=> H + CH2",  [1.08000E+14, 0, 3110])

#  Reaction 127
reaction(  "CH + H2O <=> H + CH2O",  [5.71000E+12, 0, -755])

#  Reaction 128
reaction(  "CH + CH2 <=> H + C2H2",  [4.00000E+13, 0, 0])

#  Reaction 129
reaction(  "CH + CH3 <=> H + C2H3",  [3.00000E+13, 0, 0])

#  Reaction 130
reaction(  "CH + CH4 <=> H + C2H4",  [6.00000E+13, 0, 0])

#  Reaction 131
falloff_reaction( "CH + CO (+ M) <=> HCCO (+ M)",
         kf = [5.00000E+13, 0, 0],
         kf0   = [2.69000E+28, -3.74, 1936],
         falloff = Troe(A = 0.5757, T3 = 237, T1 = 1652, T2 = 5069),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 132
```

```
reaction(  "CH + CO2 <=> HCO + CO",  [1.90000E+14, 0, 15792])

#  Reaction 133
reaction(  "CH + CH2O <=> H + CH2CO",  [9.46000E+13, 0, -515])

#  Reaction 134
reaction(  "CH + HCCO <=> CO + C2H2",  [5.00000E+13, 0, 0])

#  Reaction 135
reaction(  "CH2 + O2 => OH + H + CO",  [5.00000E+12, 0, 1500])

#  Reaction 136
reaction(  "CH2 + H2 <=> H + CH3",  [5.00000E+05, 2, 7230])

#  Reaction 137
reaction(  "2 CH2 <=> H2 + C2H2",  [1.60000E+15, 0, 11944])

#  Reaction 138
reaction(  "CH2 + CH3 <=> H + C2H4",  [4.00000E+13, 0, 0])

#  Reaction 139
reaction(  "CH2 + CH4 <=> 2 CH3",  [2.46000E+06, 2, 8270])

#  Reaction 140
falloff_reaction( "CH2 + CO (+ M) <=> CH2CO (+ M)",
         kf = [8.10000E+11, 0.5, 4510],
         kf0   = [2.69000E+33, -5.11, 7095],
         falloff = Troe(A = 0.5907, T3 = 275, T1 = 1226, T2 = 5185),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 141
reaction(  "CH2 + HCCO <=> C2H3 + CO",  [3.00000E+13, 0, 0])

#  Reaction 142
reaction(  "CH2(S) + N2 <=> CH2 + N2",  [1.50000E+13, 0, 600])

#  Reaction 143
reaction(  "CH2(S) + AR <=> CH2 + AR",  [9.00000E+12, 0, 600])

#  Reaction 144
reaction(  "CH2(S) + O2 <=> H + OH + CO",  [2.80000E+13, 0, 0])

#  Reaction 145
reaction(  "CH2(S) + O2 <=> CO + H2O",  [1.20000E+13, 0, 0])

#  Reaction 146
reaction(  "CH2(S) + H2 <=> CH3 + H",  [7.00000E+13, 0, 0])

#  Reaction 147
falloff_reaction( "CH2(S) + H2O (+ M) <=> CH3OH (+ M)",
         kf = [4.82000E+17, -1.16, 1145],
         kf0   = [1.88000E+38, -6.36, 5040],
         falloff = Troe(A = 0.6027, T3 = 208, T1 = 3922, T2 = 10180),
         efficiencies = " C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2  H2O:6 ")

#  Reaction 148
```

```
reaction(  "CH2(S) + H2O <=> CH2 + H2O",  [3.00000E+13, 0, 0])

#  Reaction 149
reaction(  "CH2(S) + CH3 <=> H + C2H4",  [1.20000E+13, 0, -570])

#  Reaction 150
reaction(  "CH2(S) + CH4 <=> 2 CH3",  [1.60000E+13, 0, -570])

#  Reaction 151
reaction(  "CH2(S) + CO <=> CH2 + CO",  [9.00000E+12, 0, 0])

#  Reaction 152
reaction(  "CH2(S) + CO2 <=> CH2 + CO2",  [7.00000E+12, 0, 0])

#  Reaction 153
reaction(  "CH2(S) + CO2 <=> CO + CH2O",  [1.40000E+13, 0, 0])

#  Reaction 154
reaction(  "CH2(S) + C2H6 <=> CH3 + C2H5",  [4.00000E+13, 0, -550])

#  Reaction 155
reaction(  "CH3 + O2 <=> O + CH3O",  [3.56000E+13, 0, 30480])

#  Reaction 156
reaction(  "CH3 + O2 <=> OH + CH2O",  [2.31000E+12, 0, 20315])

#  Reaction 157
reaction(  "CH3 + H2O2 <=> HO2 + CH4",  [2.45000E+04, 2.47, 5180])

#  Reaction 158
falloff_reaction( "2 CH3 (+ M) <=> C2H6 (+ M)",
         kf = [6.77000E+16, -1.18, 654],
         kf0   = [3.40000E+41, -7.03, 2762],
         falloff = Troe(A = 0.619, T3 = 73.2, T1 = 1180, T2 = 9999),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 159
reaction(  "2 CH3 <=> H + C2H5",  [6.84000E+12, 0.1, 10600])

#  Reaction 160
reaction(  "CH3 + HCO <=> CH4 + CO",  [2.64800E+13, 0, 0])

#  Reaction 161
reaction(  "CH3 + CH2O <=> HCO + CH4",  [3.32000E+03, 2.81, 5860])

#  Reaction 162
reaction(  "CH3 + CH3OH <=> CH2OH + CH4",  [3.00000E+07, 1.5, 9940])

#  Reaction 163
reaction(  "CH3 + CH3OH <=> CH3O + CH4",  [1.00000E+07, 1.5, 9940])

#  Reaction 164
reaction(  "CH3 + C2H4 <=> C2H3 + CH4",  [2.27000E+05, 2, 9200])

#  Reaction 165
reaction(  "CH3 + C2H6 <=> C2H5 + CH4",  [6.14000E+06, 1.74, 10450])
```

```
#  Reaction 166
reaction(  "HCO + H2O <=> H + CO + H2O",  [1.50000E+18, -1, 17000])

#  Reaction 167
three_body_reaction( "HCO + M <=> H + CO + M",  [1.87000E+17, -1,
17000],
         efficiencies = " C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2  H2O:0 ")

#  Reaction 168
reaction(  "HCO + O2 <=> HO2 + CO",  [1.34500E+13, 0, 400])

#  Reaction 169
reaction(  "CH2OH + O2 <=> HO2 + CH2O",  [1.80000E+13, 0, 900])

#  Reaction 170
reaction(  "CH3O + O2 <=> HO2 + CH2O",  [4.28000E-13, 7.6, -3530])

#  Reaction 171
reaction(  "C2H + O2 <=> HCO + CO",  [1.00000E+13, 0, -755])

#  Reaction 172
reaction(  "C2H + H2 <=> H + C2H2",  [5.68000E+10, 0.9, 1993])

#  Reaction 173
reaction(  "C2H3 + O2 <=> HCO + CH2O",  [4.58000E+16, -1.39, 1015])

#  Reaction 174
falloff_reaction( "C2H4 (+ M) <=> H2 + C2H2 (+ M)",
         kf = [8.00000E+12, 0.44, 86770],
         kf0   = [1.58000E+51, -9.3, 97800],
         falloff = Troe(A = 0.7345, T3 = 180, T1 = 1035, T2 = 5417),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 175
reaction(  "C2H5 + O2 <=> HO2 + C2H4",  [8.40000E+11, 0, 3875])

#  Reaction 176
reaction(  "HCCO + O2 <=> OH + 2 CO",  [3.20000E+12, 0, 854])

#  Reaction 177
reaction(  "2 HCCO <=> 2 CO + C2H2",  [1.00000E+13, 0, 0])

#  Reaction 178
reaction(  "N + NO <=> N2 + O",  [2.70000E+13, 0, 355])

#  Reaction 179
reaction(  "N + O2 <=> NO + O",  [9.00000E+09, 1, 6500])

#  Reaction 180
reaction(  "N + OH <=> NO + H",  [3.36000E+13, 0, 385])

#  Reaction 181
reaction(  "N2O + O <=> N2 + O2",  [1.40000E+12, 0, 10810])

#  Reaction 182
```

```
reaction(  "N2O + O <=> 2 NO",  [2.90000E+13, 0, 23150])

#  Reaction 183
reaction(  "N2O + H <=> N2 + OH",  [3.87000E+14, 0, 18880])

#  Reaction 184
reaction(  "N2O + OH <=> N2 + HO2",  [2.00000E+12, 0, 21060])

#  Reaction 185
falloff_reaction( "N2O (+ M) <=> N2 + O (+ M)",
         kf = [7.91000E+10, 0, 56020],
         kf0   = [6.37000E+14, 0, 56640],
         efficiencies = " AR:0.625  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 186
reaction(  "HO2 + NO <=> NO2 + OH",  [2.11000E+12, 0, -480])

#  Reaction 187
three_body_reaction( "NO + O + M <=> NO2 + M",  [1.06000E+20, -1.41,
0],
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 188
reaction(  "NO2 + O <=> NO + O2",  [3.90000E+12, 0, -240])

#  Reaction 189
reaction(  "NO2 + H <=> NO + OH",  [1.32000E+14, 0, 360])

#  Reaction 190
reaction(  "NH + O <=> NO + H",  [4.00000E+13, 0, 0])

#  Reaction 191
reaction(  "NH + H <=> N + H2",  [3.20000E+13, 0, 330])

#  Reaction 192
reaction(  "NH + OH <=> HNO + H",  [2.00000E+13, 0, 0])

#  Reaction 193
reaction(  "NH + OH <=> N + H2O",  [2.00000E+09, 1.2, 0])

#  Reaction 194
reaction(  "NH + O2 <=> HNO + O",  [4.61000E+05, 2, 6500])

#  Reaction 195
reaction(  "NH + O2 <=> NO + OH",  [1.28000E+06, 1.5, 100])

#  Reaction 196
reaction(  "NH + N <=> N2 + H",  [1.50000E+13, 0, 0])

#  Reaction 197
reaction(  "NH + H2O <=> HNO + H2",  [2.00000E+13, 0, 13850])

#  Reaction 198
reaction(  "NH + NO <=> N2 + OH",  [2.16000E+13, -0.23, 0])
```

```
#  Reaction 199
reaction(  "NH + NO <=> N2O + H",  [3.65000E+14, -0.45, 0])

#  Reaction 200
reaction(  "NH2 + O <=> OH + NH",  [3.00000E+12, 0, 0])

#  Reaction 201
reaction(  "NH2 + O <=> H + HNO",  [3.90000E+13, 0, 0])

#  Reaction 202
reaction(  "NH2 + H <=> NH + H2",  [4.00000E+13, 0, 3650])

#  Reaction 203
reaction(  "NH2 + OH <=> NH + H2O",  [9.00000E+07, 1.5, -460])

#  Reaction 204
reaction(  "NNH <=> N2 + H",  [3.30000E+08, 0, 0])

#  Reaction 205
three_body_reaction( "NNH + M <=> N2 + H + M",  [1.30000E+14, -0.11,
4980],
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 206
reaction(  "NNH + O2 <=> HO2 + N2",  [5.00000E+12, 0, 0])

#  Reaction 207
reaction(  "NNH + O <=> OH + N2",  [2.50000E+13, 0, 0])

#  Reaction 208
reaction(  "NNH + O <=> NH + NO",  [7.00000E+13, 0, 0])

#  Reaction 209
reaction(  "NNH + H <=> H2 + N2",  [5.00000E+13, 0, 0])

#  Reaction 210
reaction(  "NNH + OH <=> H2O + N2",  [2.00000E+13, 0, 0])

#  Reaction 211
reaction(  "NNH + CH3 <=> CH4 + N2",  [2.50000E+13, 0, 0])

#  Reaction 212
three_body_reaction( "H + NO + M <=> HNO + M",  [4.48000E+19, -1.32,
740],
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 213
reaction(  "HNO + O <=> NO + OH",  [2.50000E+13, 0, 0])

#  Reaction 214
reaction(  "HNO + H <=> H2 + NO",  [9.00000E+11, 0.72, 660])

#  Reaction 215
reaction(  "HNO + OH <=> NO + H2O",  [1.30000E+07, 1.9, -950])
```

```
#  Reaction 216
reaction(  "HNO + O2 <=> HO2 + NO",  [1.00000E+13, 0, 13000])

#  Reaction 217
reaction(  "CN + O <=> CO + N",  [7.70000E+13, 0, 0])

#  Reaction 218
reaction(  "CN + OH <=> NCO + H",  [4.00000E+13, 0, 0])

#  Reaction 219
reaction(  "CN + H2O <=> HCN + OH",  [8.00000E+12, 0, 7460])

#  Reaction 220
reaction(  "CN + O2 <=> NCO + O",  [6.14000E+12, 0, -440])

#  Reaction 221
reaction(  "CN + H2 <=> HCN + H",  [2.95000E+05, 2.45, 2240])

#  Reaction 222
reaction(  "NCO + O <=> NO + CO",  [2.35000E+13, 0, 0])

#  Reaction 223
reaction(  "NCO + H <=> NH + CO",  [5.40000E+13, 0, 0])

#  Reaction 224
reaction(  "NCO + OH <=> NO + H + CO",  [2.50000E+12, 0, 0])

#  Reaction 225
reaction(  "NCO + N <=> N2 + CO",  [2.00000E+13, 0, 0])

#  Reaction 226
reaction(  "NCO + O2 <=> NO + CO2",  [2.00000E+12, 0, 20000])

#  Reaction 227
three_body_reaction( "NCO + M <=> N + CO + M",  [3.10000E+14, 0,
54050],
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 228
reaction(  "NCO + NO <=> N2O + CO",  [1.90000E+17, -1.52, 740])

#  Reaction 229
reaction(  "NCO + NO <=> N2 + CO2",  [3.80000E+18, -2, 800])

#  Reaction 230
three_body_reaction( "HCN + M <=> H + CN + M",  [1.04000E+29, -3.3,
126600],
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 231
reaction(  "HCN + O <=> NCO + H",  [2.03000E+04, 2.64, 4980])

#  Reaction 232
reaction(  "HCN + O <=> NH + CO",  [5.07000E+03, 2.64, 4980])
```

```
#  Reaction 233
reaction(  "HCN + O <=> CN + OH",  [3.91000E+09, 1.58, 26600])

#  Reaction 234
reaction(  "HCN + OH <=> HOCN + H",  [1.10000E+06, 2.03, 13370])

#  Reaction 235
reaction(  "HCN + OH <=> HNCO + H",  [4.40000E+03, 2.26, 6400])

#  Reaction 236
reaction(  "HCN + OH <=> NH2 + CO",  [1.60000E+02, 2.56, 9000])

#  Reaction 237
falloff_reaction( "H + HCN (+ M) <=> H2CN (+ M)",
         kf = [3.30000E+13, 0, 0],
         kf0   = [1.40000E+26, -3.4, 1900],
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 238
reaction(  "H2CN + N <=> N2 + CH2",  [6.00000E+13, 0, 400])

#  Reaction 239
reaction(  "C + N2 <=> CN + N",  [6.30000E+13, 0, 46020])

#  Reaction 240
reaction(  "CH + N2 <=> HCN + N",  [3.12000E+09, 0.88, 20130])

#  Reaction 241
falloff_reaction( "CH + N2 (+ M) <=> HCNN (+ M)",
         kf = [3.10000E+12, 0.15, 0],
         kf0   = [1.30000E+25, -3.16, 740],
         falloff = Troe(A = 0.667, T3 = 235, T1 = 2117, T2 = 4536),
         efficiencies = " AR:1  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 242
reaction(  "CH2 + N2 <=> HCN + NH",  [1.00000E+13, 0, 74000])

#  Reaction 243
reaction(  "CH2(S) + N2 <=> NH + HCN",  [1.00000E+11, 0, 65000])

#  Reaction 244
reaction(  "C + NO <=> CN + O",  [1.90000E+13, 0, 0])

#  Reaction 245
reaction(  "C + NO <=> CO + N",  [2.90000E+13, 0, 0])

#  Reaction 246
reaction(  "CH + NO <=> HCN + O",  [4.10000E+13, 0, 0])

#  Reaction 247
reaction(  "CH + NO <=> H + NCO",  [1.62000E+13, 0, 0])

#  Reaction 248
reaction(  "CH + NO <=> N + HCO",  [2.46000E+13, 0, 0])
```

```
#  Reaction 249
reaction(  "CH2 + NO <=> H + HNCO",  [3.10000E+17, -1.38, 1270])

#  Reaction 250
reaction(  "CH2 + NO <=> OH + HCN",  [2.90000E+14, -0.69, 760])

#  Reaction 251
reaction(  "CH2 + NO <=> H + HCNO",  [3.80000E+13, -0.36, 580])

#  Reaction 252
reaction(  "CH2(S) + NO <=> H + HNCO",  [3.10000E+17, -1.38, 1270])

#  Reaction 253
reaction(  "CH2(S) + NO <=> OH + HCN",  [2.90000E+14, -0.69, 760])

#  Reaction 254
reaction(  "CH2(S) + NO <=> H + HCNO",  [3.80000E+13, -0.36, 580])

#  Reaction 255
reaction(  "CH3 + NO <=> HCN + H2O",  [9.60000E+13, 0, 28800])

#  Reaction 256
reaction(  "CH3 + NO <=> H2CN + OH",  [1.00000E+12, 0, 21750])

#  Reaction 257
reaction(  "HCNN + O <=> CO + H + N2",  [2.20000E+13, 0, 0])

#  Reaction 258
reaction(  "HCNN + O <=> HCN + NO",  [2.00000E+12, 0, 0])

#  Reaction 259
reaction(  "HCNN + O2 <=> O + HCO + N2",  [1.20000E+13, 0, 0])

#  Reaction 260
reaction(  "HCNN + OH <=> H + HCO + N2",  [1.20000E+13, 0, 0])

#  Reaction 261
reaction(  "HCNN + H <=> CH2 + N2",  [1.00000E+14, 0, 0])

#  Reaction 262
reaction(  "HNCO + O <=> NH + CO2",  [9.80000E+07, 1.41, 8500])

#  Reaction 263
reaction(  "HNCO + O <=> HNO + CO",  [1.50000E+08, 1.57, 44000])

#  Reaction 264
reaction(  "HNCO + O <=> NCO + OH",  [2.20000E+06, 2.11, 11400])

#  Reaction 265
reaction(  "HNCO + H <=> NH2 + CO",  [2.25000E+07, 1.7, 3800])

#  Reaction 266
reaction(  "HNCO + H <=> H2 + NCO",  [1.05000E+05, 2.5, 13300])

#  Reaction 267
reaction(  "HNCO + OH <=> NCO + H2O",  [3.30000E+07, 1.5, 3600])
```

```
#  Reaction 268
reaction(  "HNCO + OH <=> NH2 + CO2",  [3.30000E+06, 1.5, 3600])

#  Reaction 269
three_body_reaction( "HNCO + M <=> NH + CO + M",  [1.18000E+16, 0,
84720],
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 270
reaction(  "HCNO + H <=> H + HNCO",  [2.10000E+15, -0.69, 2850])

#  Reaction 271
reaction(  "HCNO + H <=> OH + HCN",  [2.70000E+11, 0.18, 2120])

#  Reaction 272
reaction(  "HCNO + H <=> NH2 + CO",  [1.70000E+14, -0.75, 2890])

#  Reaction 273
reaction(  "HOCN + H <=> H + HNCO",  [2.00000E+07, 2, 2000])

#  Reaction 274
reaction(  "HCCO + NO <=> HCNO + CO",  [9.00000E+12, 0, 0])

#  Reaction 275
reaction(  "CH3 + N <=> H2CN + H",  [6.10000E+14, -0.31, 290])

#  Reaction 276
reaction(  "CH3 + N <=> HCN + H2",  [3.70000E+12, 0.15, -90])

#  Reaction 277
reaction(  "NH3 + H <=> NH2 + H2",  [5.40000E+05, 2.4, 9915])

#  Reaction 278
reaction(  "NH3 + OH <=> NH2 + H2O",  [5.00000E+07, 1.6, 955])

#  Reaction 279
reaction(  "NH3 + O <=> NH2 + OH",  [9.40000E+06, 1.94, 6460])

#  Reaction 280
reaction(  "NH + CO2 <=> HNO + CO",  [1.00000E+13, 0, 14350])

#  Reaction 281
reaction(  "CN + NO2 <=> NCO + NO",  [6.16000E+15, -0.752, 345])

#  Reaction 282
reaction(  "NCO + NO2 <=> N2O + CO2",  [3.25000E+12, 0, -705])

#  Reaction 283
reaction(  "N + CO2 <=> NO + CO",  [3.00000E+12, 0, 11300])

#  Reaction 284
reaction(  "O + CH3 => H + H2 + CO",  [3.37000E+13, 0, 0])

#  Reaction 285
reaction(  "O + C2H4 <=> H + CH2CHO",  [6.70000E+06, 1.83, 220])
```

```
#  Reaction 286
reaction(  "O + C2H5 <=> H + CH3CHO",  [1.09600E+14, 0, 0])

#  Reaction 287
reaction(  "OH + HO2 <=> O2 + H2O",  [5.00000E+15, 0, 17330],
         options = ["duplicate"])

#  Reaction 288
reaction(  "OH + CH3 => H2 + CH2O",  [8.00000E+09, 0.5, -1755])

#  Reaction 289
falloff_reaction( "CH + H2 (+ M) <=> CH3 (+ M)",
         kf = [1.97000E+12, 0.43, -370],
         kf0   = [4.82000E+25, -2.8, 590],
         falloff = Troe(A = 0.578, T3 = 122, T1 = 2535, T2 = 9365),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 290
reaction(  "CH2 + O2 => 2 H + CO2",  [5.80000E+12, 0, 1500])

#  Reaction 291
reaction(  "CH2 + O2 <=> O + CH2O",  [2.40000E+12, 0, 1500])

#  Reaction 292
reaction(  "CH2 + CH2 => 2 H + C2H2",  [2.00000E+14, 0, 10989])

#  Reaction 293
reaction(  "CH2(S) + H2O => H2 + CH2O",  [6.82000E+10, 0.25, -935])

#  Reaction 294
reaction(  "C2H3 + O2 <=> O + CH2CHO",  [3.03000E+11, 0.29, 11])

#  Reaction 295
reaction(  "C2H3 + O2 <=> HO2 + C2H2",  [1.33700E+06, 1.61, -384])

#  Reaction 296
reaction(  "O + CH3CHO <=> OH + CH2CHO",  [2.92000E+12, 0, 1808])

#  Reaction 297
reaction(  "O + CH3CHO => OH + CH3 + CO",  [2.92000E+12, 0, 1808])

#  Reaction 298
reaction(  "O2 + CH3CHO => HO2 + CH3 + CO",  [3.01000E+13, 0, 39150])

#  Reaction 299
reaction(  "H + CH3CHO <=> CH2CHO + H2",  [2.05000E+09, 1.16, 2405])

#  Reaction 300
reaction(  "H + CH3CHO => CH3 + H2 + CO",  [2.05000E+09, 1.16, 2405])

#  Reaction 301
reaction(  "OH + CH3CHO => CH3 + H2O + CO",  [2.34300E+10, 0.73, -
1113])

#  Reaction 302
reaction(  "HO2 + CH3CHO => CH3 + H2O2 + CO",  [3.01000E+12, 0, 11923])
```

```
#  Reaction 303
reaction(  "CH3 + CH3CHO => CH3 + CH4 + CO",  [2.72000E+06, 1.77,
5920])

#  Reaction 304
falloff_reaction( "H + CH2CO (+ M) <=> CH2CHO (+ M)",
         kf = [4.86500E+11, 0.422, -1755],
         kf0   = [1.01200E+42, -7.63, 3854],
         falloff = Troe(A = 0.465, T3 = 201, T1 = 1773, T2 = 5333),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 305
reaction(  "O + CH2CHO => H + CH2 + CO2",  [1.50000E+14, 0, 0])

#  Reaction 306
reaction(  "O2 + CH2CHO => OH + CO + CH2O",  [1.81000E+10, 0, 0])

#  Reaction 307
reaction(  "O2 + CH2CHO => OH + 2 HCO",  [2.35000E+10, 0, 0])

#  Reaction 308
reaction(  "H + CH2CHO <=> CH3 + HCO",  [2.20000E+13, 0, 0])

#  Reaction 309
reaction(  "H + CH2CHO <=> CH2CO + H2",  [1.10000E+13, 0, 0])

#  Reaction 310
reaction(  "OH + CH2CHO <=> H2O + CH2CO",  [1.20000E+13, 0, 0])

#  Reaction 311
reaction(  "OH + CH2CHO <=> HCO + CH2OH",  [3.01000E+13, 0, 0])

#  Reaction 312
falloff_reaction( "CH3 + C2H5 (+ M) <=> C3H8 (+ M)",
         kf = [9.43000E+12, 0, 0],
         kf0   = [2.71000E+74, -16.82, 13065],
         falloff = Troe(A = 0.1527, T3 = 291, T1 = 2742, T2 = 7748),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 313
reaction(  "O + C3H8 <=> OH + C3H7",  [1.93000E+05, 2.68, 3716])

#  Reaction 314
reaction(  "H + C3H8 <=> C3H7 + H2",  [1.32000E+06, 2.54, 6756])

#  Reaction 315
reaction(  "OH + C3H8 <=> C3H7 + H2O",  [3.16000E+07, 1.8, 934])

#  Reaction 316
reaction(  "C3H7 + H2O2 <=> HO2 + C3H8",  [3.78000E+02, 2.72, 1500])

#  Reaction 317
reaction(  "CH3 + C3H8 <=> C3H7 + CH4",  [9.03000E-01, 3.65, 7154])
```

```
#  Reaction 318
falloff_reaction( "CH3 + C2H4 (+ M) <=> C3H7 (+ M)",
         kf = [2.55000E+06, 1.6, 5700],
         kf0   = [3.00000E+63, -14.6, 18170],
         falloff = Troe(A = 0.1894, T3 = 277, T1 = 8748, T2 = 7891),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 319
reaction(  "O + C3H7 <=> C2H5 + CH2O",  [9.64000E+13, 0, 0])

#  Reaction 320
falloff_reaction( "H + C3H7 (+ M) <=> C3H8 (+ M)",
         kf = [3.61300E+13, 0, 0],
         kf0   = [4.42000E+61, -13.545, 11357],
         falloff = Troe(A = 0.315, T3 = 369, T1 = 3285, T2 = 6667),
         efficiencies = " AR:0.7  C2H6:3  CH4:2  CO:1.5  CO2:2  H2:2
H2O:6 ")

#  Reaction 321
reaction(  "H + C3H7 <=> CH3 + C2H5",  [4.06000E+06, 2.19, 890])

#  Reaction 322
reaction(  "OH + C3H7 <=> C2H5 + CH2OH",  [2.41000E+13, 0, 0])

#  Reaction 323
reaction(  "HO2 + C3H7 <=> O2 + C3H8",  [2.55000E+10, 0.255, -943])

#  Reaction 324
reaction(  "HO2 + C3H7 => OH + C2H5 + CH2O",  [2.41000E+13, 0, 0])

#  Reaction 325
reaction(  "CH3 + C3H7 <=> 2 C2H5",  [1.92700E+13, -0.32, 0])

#  Charged Reaction 1 (methane_ion31.cti reaction 21)
reaction( "CH + O => HCO+ + E",   [5.75000E+11, 0, 6000])

#  Charged Reaction 2 (methane_ion31.cti reaction 22)
reaction( "HCO+ + H2O => CO + H3O+",   [5.02000E+17, 0, 24000])

#  Charged Reaction 3 (methane_ion31.cti reaction 23)
reaction( "H3O+ + E => H2O + H",   [1.44000E+17, 0, 0])
```

B-3: Listing of Pederson and Brown Mechanism

```
#
# Generated from file PB93.che
# by ck2cti on Tue Jul 26 11:40:13 2005
#
# Transport data from file PB93_trans.dat.

units(length = "cm", time = "s", quantity = "mol", act_energy =
"cal/mol")


#      species = """ C2H  C2H2  C2H3  C2H3O+  C2H4  C2H5  C2H6  C3H3
C3H3+  CH
#                  CH*  CH2  CH2O  CH3  CH3+  CH4  CHO  CHO+  CO  CO2
#                  H  H2  H2O  H3O+  HO2  O  O2  OH  E  N
#                  N2 """,
ideal_gas(name = "gas",
      elements = " O  H  C  E  N ",
      species = """ C2H  C2H2  C2H3  C2H4  C2H5  C2H6  C3H3  C3H3+  CH
                    CH*  CH2  CH2O  CH3  CH3+  CH4  CHO  CHO+  CO  CO2
                    H  H2  H2O  H3O+  HO2  O  O2  OH  E  N
                    N2 """,
      reactions = "all",
      transport = "Mix",
      initial_state = state(temperature = 300.0,
                        pressure = OneAtm)    )



#-----------------------------------------------------------------------
---------
#  Species data
#-----------------------------------------------------------------------
---------

species(name = "C2H",
    atoms = " C:2  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.889657330E+00,   1.340996110E-
02,
                -2.847695010E-05,   2.947910450E-08,  -1.093315110E-11,
                 6.683939320E+04,   6.222964380E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.167806520E+00,   4.752219020E-
03,
                -1.837870770E-06,   3.041902520E-10,  -1.772327700E-14,
                 6.712106500E+04,   6.635894750E+00] )
             ),
    transport = gas_transport(
                      geom = "linear",
                      diam =     4.10,
                      well_depth =   209.00,
                      rot_relax =     2.50),
    note = "L 1/91"
       )
```

```
species(name = "C2H2",
    atoms = " C:2  H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  8.086810940E-01,   2.336156290E-
02,
              -3.551718150E-05,   2.801524370E-08,  -8.500729740E-12,
               2.642898070E+04,   1.393970510E+01] ),
       NASA( [ 1000.00,  3500.00], [  4.147569640E+00,   5.961666640E-
03,
              -2.372948520E-06,   4.674121710E-10,  -3.612352130E-14,
               2.593599920E+04,  -1.230281210E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     4.10,
                     well_depth =   209.00,
                     rot_relax =     2.50),
    note = "L 1/91"
       )

species(name = "C2H3",
    atoms = " C:2  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.212466450E+00,   1.514791620E-
03,
               2.592094120E-05,  -3.576578470E-08,   1.471508730E-11,
               3.485984680E+04,   8.510540250E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.016724000E+00,   1.033022920E-
02,
              -4.680823490E-06,   1.017632880E-09,  -8.626070410E-14,
               3.461287390E+04,   7.787323780E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.10,
                     well_depth =   209.00,
                     rot_relax =     1.00),
    note = "L 2/92"
       )

#species(name = "C2H3O+",
#    atoms = " C:2  H:3  O:1  E:-1 ",
#    thermo = (
#       NASA( [  298.15,  1000.00], [  1.085477200E+00,   1.284525900E-
02,
#              2.413866000E-06,  -4.464267200E-09,  -2.938191600E-13,
#              1.591065500E+04,   3.339531200E+01] ),
#       NASA( [ 1000.00,  3000.00], [  4.813147000E-01,   2.071191400E-
02,
#             -1.269315500E-05,   3.457964200E-09,  -3.539970300E-13,
#              1.564864200E+04,   3.462987600E+01] )
#             ),
#    transport = gas_transport(
#                     geom = "nonlinear",
#                     diam =     3.97,
#                     well_depth =   436.00,
```

```
#                      rot_relax =     2.00),
#    note = "T 9/92"
#       )

species(name = "C2H4",
    atoms = " C:2  H:4 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.959201480E+00,  -7.570522470E-
03,
                5.709902920E-05,  -6.915887530E-08,   2.698843730E-11,
                5.089775930E+03,   4.097330960E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.036111160E+00,   1.464541510E-
02,
               -6.710779150E-06,   1.472229230E-09,  -1.257060610E-13,
                4.939886140E+03,   1.030536930E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.97,
                     well_depth =   280.80,
                     rot_relax =     1.50),
    note = "L 1/91"
       )

species(name = "C2H5",
    atoms = " C:2  H:5 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.306465680E+00,  -4.186588920E-
03,
                4.971428070E-05,  -5.991266060E-08,   2.305090040E-11,
                1.284162650E+04,   4.707209240E+00] ),
       NASA( [ 1000.00,  3500.00], [  1.954656420E+00,   1.739727220E-
02,
               -7.982066680E-06,   1.752176890E-09,  -1.496415760E-13,
                1.285752000E+04,   1.346243430E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.30,
                     well_depth =   252.30,
                     rot_relax =     1.50),
    note = "L12/92"
       )

species(name = "C2H6",
    atoms = " C:2  H:6 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.291424920E+00,  -5.501542700E-
03,
                5.994382880E-05,  -7.084662850E-08,   2.686857710E-11,
               -1.152220550E+04,   2.666823160E+00] ),
       NASA( [ 1000.00,  3500.00], [  1.071881500E+00,   2.168526770E-
02,
               -1.002560670E-05,   2.214120010E-09,  -1.900028900E-13,
               -1.142639320E+04,   1.511561070E+01] )
             ),
    transport = gas_transport(
```

```
                    geom = "nonlinear",
                    diam =     4.30,
                    well_depth =   252.30,
                    rot_relax =     1.50),
    note = "L 8/88"
       )

species(name = "C3H3",
    atoms = " C:3  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  1.828407660E+00,   2.378390360E-
02,
              -2.192281760E-05,   1.000674440E-08,  -1.389846440E-12,
               4.018630580E+04,   1.384479570E+01] ),
       NASA( [ 1000.00,  6000.00], [  6.641758210E+00,   8.085874280E-
03,
              -2.847878870E-06,   4.535259770E-10,  -2.688798150E-14,
               3.897936990E+04,  -1.040042550E+01] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     4.76,
                    well_depth =   252.00,
                    rot_relax =     1.00),
    note = "BUR 92"
       )

species(name = "C3H3+",
    atoms = " C:3  H:3  E:-1 ",
    thermo = const_cp(t0 = 298.15,
               h0 = (255.0, 'cal/mol'),
               s0 = (59.0, 'cal/K/mol'),
               cp0 = (14.1, 'cal/K/mol')),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     4.76,
                    well_depth =   252.00,
                    rot_relax =     1.00),
    note = "REPLAC"
       )

species(name = "CH",
    atoms = " C:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.489816650E+00,   3.238355410E-
04,
              -1.688990650E-06,   3.162173270E-09,  -1.406090670E-12,
               7.079729340E+04,   2.084011080E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.878464730E+00,   9.709136810E-
04,
               1.444456550E-07,  -1.306878490E-10,   1.760793830E-14,
               7.101243640E+04,   5.484979990E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     2.75,
                    well_depth =    80.00),
```

```
    note = "TPIS79"
       )

species(name = "CH*",
    atoms = " C:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.489816650E+00,   3.238355410E-
04,
               -1.688990650E-06,   3.162173270E-09,  -1.406090670E-12,
                2.776707973E+08,   2.084011080E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.878464730E+00,   9.709136810E-
04,
                1.444456550E-07,  -1.306878490E-10,   1.760793830E-14,
                2.776710124E+08,   5.484979990E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "TPIS79"
       )

species(name = "CH2",
    atoms = " C:1  H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.762678670E+00,   9.688721430E-
04,
                2.794898410E-06,  -3.850911530E-09,   1.687417190E-12,
                4.600404010E+04,   1.562531850E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.874101130E+00,   3.656392920E-
03,
               -1.408945970E-06,   2.601795490E-10,  -1.877275670E-14,
                4.626360400E+04,   6.171193240E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.80,
                     well_depth =   144.00),
    note = "L S/93"
       )

species(name = "CH2O",
    atoms = " H:2  C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.793723150E+00,  -9.908333690E-
03,
                3.732200080E-05,  -3.792852610E-08,   1.317726520E-11,
               -1.430895670E+04,   6.028129000E-01] ),
       NASA( [ 1000.00,  3500.00], [  1.760690080E+00,   9.200000820E-
03,
               -4.422588130E-06,   1.006412120E-09,  -8.838556400E-14,
               -1.399583230E+04,   1.365632300E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00,
```

```
                    rot_relax =     2.00),
    note = "L 8/88"
       )

species(name = "CH3",
    atoms = " C:1  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.673590400E+00,   2.010951750E-
03,
                5.730218560E-06,  -6.871174250E-09,   2.543857340E-12,
                1.644499880E+04,   1.604564330E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.285717720E+00,   7.239900370E-
03,
               -2.987143480E-06,   5.956846440E-10,  -4.671543940E-14,
                1.677558430E+04,   8.480071790E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.80,
                    well_depth =   144.00),
    note = "L11/89"
       )

species(name = "CH3+",
    atoms = " C:1  H:3  E:-1 ",
    thermo = const_cp(t0 = 298.15,
                  h0 = (260.0, 'cal/mol'),
                  s0 = (46.7, 'cal/K/mol'),
                  cp0 = (9.3, 'cal/K/mol')),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.80,
                    well_depth =   144.00),
    note = "REPLAC"
       )

species(name = "CH4",
    atoms = " C:1  H:4 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  5.149876130E+00,  -1.367097880E-
02,
                4.918005990E-05,  -4.847430260E-08,   1.666939560E-11,
               -1.024664760E+04,  -4.641303760E+00] ),
       NASA( [ 1000.00,  3500.00], [  7.485149500E-02,   1.339094670E-
02,
               -5.732858090E-06,   1.222925350E-09,  -1.018152300E-13,
               -9.468344590E+03,   1.843731800E+01] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     3.75,
                    well_depth =   141.40,
                    polar =     2.60,
                    rot_relax =    13.00),
    note = "L 8/88"
       )
```

```
species(name = "CHO",
    atoms = " C:1  H:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.237546100E+00,  -3.320752570E-
03,
                1.400302640E-05,  -1.342399950E-08,   4.374162080E-12,
                3.872411850E+03,   3.308348690E+00] ),
       NASA( [ 1000.00,  6000.00], [  3.920015420E+00,   2.522793240E-
03,
               -6.710041640E-07,   1.056159480E-10,  -7.437982610E-15,
                3.653429280E+03,   3.580770560E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00),
    note = "T 5/03"
       )

species(name = "CHO+",
    atoms = " H:1  C:1  O:1  E:-1 ",
    thermo = const_cp(t0 = 298.15,
                  h0 = (199.1, 'cal/mol'),
                  s0 = (46.7, 'cal/K/mol'),
                  cp0 = (8.6, 'cal/K/mol')),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00),
    note = "REPLAC"
       )

species(name = "CO",
    atoms = " C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.579533470E+00,  -6.103536800E-
04,
                1.016814330E-06,   9.070058840E-10,  -9.044244990E-13,
               -1.434408600E+04,   3.508409280E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.715185610E+00,   2.062527430E-
03,
               -9.988257710E-07,   2.300530080E-10,  -2.036477160E-14,
               -1.415187240E+04,   7.818687720E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.65,
                     well_depth =    98.10,
                     polar =     1.95,
                     rot_relax =     1.80),
    note = "TPIS79"
       )

species(name = "CO2",
    atoms = " C:1  O:2 ",
    thermo = (
```

```
       NASA( [  200.00,  1000.00], [  2.356773520E+00,   8.984596770E-
03,
               -7.123562690E-06,   2.459190220E-09,  -1.436995480E-13,
               -4.837196970E+04,   9.901052220E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.857460290E+00,   4.414370260E-
03,
               -2.214814040E-06,   5.234901880E-10,  -4.720841640E-14,
               -4.875916600E+04,   2.271638060E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.76,
                    well_depth =   244.00,
                    polar =     2.65,
                    rot_relax =     2.10),
    note = "L 7/88"
       )

species(name = "H",
    atoms = " H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.500000000E+00,   7.053328190E-
13,
               -1.995919640E-15,   2.300816320E-18,  -9.277323320E-22,
                2.547365990E+04,  -4.466828530E-01] ),
       NASA( [ 1000.00,  3500.00], [  2.500000010E+00,  -2.308429730E-
11,
                1.615619480E-14,  -4.735152350E-18,   4.981973570E-22,
                2.547365990E+04,  -4.466829140E-01] )
             ),
    transport = gas_transport(
                    geom = "atom",
                    diam =     2.05,
                    well_depth =   145.00),
    note = "L 7/88"
       )

species(name = "H2",
    atoms = " H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.344331120E+00,   7.980520750E-
03,
               -1.947815100E-05,   2.015720940E-08,  -7.376117610E-12,
               -9.179351730E+02,   6.830102380E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.337279200E+00,  -4.940247310E-
05,
                4.994567780E-07,  -1.795663940E-10,   2.002553760E-14,
               -9.501589220E+02,  -3.205023310E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     2.92,
                    well_depth =    38.00,
                    polar =     0.79,
                    rot_relax =   280.00),
    note = "TPIS78"
       )
```

```
species(name = "H2O",
    atoms = " H:2  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.198640560E+00,  -2.036434100E-
03,
                6.520402110E-06,  -5.487970620E-09,   1.771978170E-12,
               -3.029372670E+04,  -8.490322080E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.033992490E+00,   2.176918040E-
03,
               -1.640725180E-07,  -9.704198700E-11,   1.682009920E-14,
               -3.000429710E+04,   4.966770100E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.60,
                     well_depth =   572.40,
                     dipole =     1.84,
                     rot_relax =     4.00),
    note = "L 8/89"
       )

species(name = "H3O+",
    atoms = " H:3  O:1  E:-1 ",
    thermo = const_cp(t0 = 298.15,
                  h0 = (139.0, 'cal/mol'),
                  s0 = (46.7, 'cal/K/mol'),
                  cp0 = (8.4, 'cal/K/mol')),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.60,
                     well_depth =   572.40,
                     dipole =     1.84,
                     rot_relax =     4.00),
    note = "REPLAC"
       )

species(name = "HO2",
    atoms = " H:1  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.301798010E+00,  -4.749120510E-
03,
                2.115828910E-05,  -2.427638940E-08,   9.292251240E-12,
                2.948080400E+02,   3.716662450E+00] ),
       NASA( [ 1000.00,  3500.00], [  4.017210900E+00,   2.239820130E-
03,
               -6.336581500E-07,   1.142463700E-10,  -1.079085350E-14,
                1.118567130E+02,   3.785102150E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.46,
                     well_depth =   107.40,
                     rot_relax =     1.00),
    note = "L 5/89"
       )
```

```
species(name = "O",
    atoms = " O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.168267100E+00,  -3.279318840E-
03,
                6.643063960E-06,  -6.128066240E-09,   2.112659710E-12,
                2.912225920E+04,   2.051933460E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.569420780E+00,  -8.597411370E-
05,
                4.194845890E-08,  -1.001777990E-11,   1.228336910E-15,
                2.921757910E+04,   4.784338640E+00] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "L 1/90"
       )

species(name = "O2",
    atoms = " O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.782456360E+00,  -2.996734160E-
03,
                9.847302010E-06,  -9.681295090E-09,   3.243728370E-12,
               -1.063943560E+03,   3.657675730E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.282537840E+00,   1.483087540E-
03,
               -7.579666690E-07,   2.094705550E-10,  -2.167177940E-14,
               -1.088457720E+03,   5.453231290E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.46,
                     well_depth =   107.40,
                     polar =     1.60,
                     rot_relax =     3.80),
    note = "TPIS89"
       )

species(name = "OH",
    atoms = " O:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.992015430E+00,  -2.401317520E-
03,
                4.617938410E-06,  -3.881133330E-09,   1.364114700E-12,
                3.615080560E+03,  -1.039254580E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.092887670E+00,   5.484297160E-
04,
                1.265052280E-07,  -8.794615560E-11,   1.174123760E-14,
                3.858657000E+03,   4.476696100E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "RUS 78"
```

```
       )

species(name = "E",
    atoms = " E:1 ",
    thermo = const_cp(t0 = 298.15,
                  h0 = (0.0, 'cal/mol'),
                  s0 = (4.9, 'cal/K/mol'),
                  cp0 = (4.9, 'cal/K/mol')),
    transport = gas_transport(
                     geom = "atom",
                     diam =   425.00,
                     well_depth =   850.00,
                     rot_relax =     1.00),
    note = "REPLAC"
       )

species(name = "N",
    atoms = " N:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
                5.610463700E+04,   4.193908700E+00] ),
       NASA( [ 1000.00,  6000.00], [  2.415942900E+00,   1.748906500E-
04,
               -1.190236900E-07,   3.022624500E-11,  -2.036098200E-15,
                5.613377300E+04,   4.649609600E+00] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     3.30,
                     well_depth =    71.40),
    note = "L 6/88"
       )

species(name = "N2",
    atoms = " N:2 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  3.298677000E+00,   1.408240400E-
03,
               -3.963222000E-06,   5.641515000E-09,  -2.444854000E-12,
               -1.020899900E+03,   3.950372000E+00] ),
       NASA( [ 1000.00,  5000.00], [  2.926640000E+00,   1.487976800E-
03,
               -5.684760000E-07,   1.009703800E-10,  -6.753351000E-15,
               -9.227977000E+02,   5.980528000E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.62,
                     well_depth =    97.53,
                     polar =     1.76,
                     rot_relax =     4.00),
    note = "121286"
       )
```

```
#-----------------------------------------------------------------------
---------
#  Reaction data
#-----------------------------------------------------------------------
---------

#  Reaction 1
# 1
reaction(  "OH + H2 => H2O + H",  [1.17000E+09, 1.3, 1825])

#  Reaction 2
# 2
#
reaction(  "H + O2 => OH + O",  [1.42000E+14, 0, 8250])

#  Reaction 3
# 3
#
reaction(  "O + H2 => OH + H",  [1.80000E+10, 1, 4480])

#  Reaction 4
# 4
#  M'
three_body_reaction( "H + O2 + M => HO2 + M",  [1.03000E+18, -0.72, 0])

#  Reaction 5
# 5
#
reaction(  "H + HO2 => OH + OH",  [1.40000E+14, 0, 540])

#  Reaction 6
# 6
#
reaction(  "H + HO2 => O + H2O",  [1.00000E+13, 0, 540])

#  Reaction 7
# 7
#
reaction(  "HO2 + H => O2 + H2",  [1.25000E+13, 0, 0])

#  Reaction 8
# 8
#
reaction(  "HO2 + OH => O2 + H2O",  [7.50000E+12, 0, 0])

#  Reaction 9
# 9
#
reaction(  "HO2 + O => O2 + OH",  [1.40000E+13, 0, 540])

#  Reaction 10
# 10
#
reaction(  "H + H + H2 => H2 + H2",  [9.20000E+16, -0.60, 0])

#  Reaction 11
```

```
# 11
#
reaction(  "H + H + O2 => H2 + O2",  [1.00000E+18, -1, 0])

#  Reaction 12
# 12
#
reaction(  "H + H + H2O => H2 + H2O",  [6.00000E+19, -1.25, 0])

#  Reaction 13
# 13
#
reaction(  "H + H + CO => H2 + CO",  [1.00000E+18, -1, 0])

#  Reaction 14
# 14
#
reaction(  "H + H + CO2 => H2 + CO2",  [5.49000E+20, -2, 0])

#  Reaction 15
# 15
#
reaction(  "H + H + CH4 => H2 + CH4",  [5.49000E+20, -2, 0])

#  Reaction 16
# 16
#  M"
three_body_reaction( "H + OH + M => H2O + M",  [1.60000E+22, -2, 0])

#  Reaction 17
# 17
#  M"
three_body_reaction( "H + O + M => OH + M",  [6.20000E+16, -0.60, 0])

#  Reaction 18
# 18
#
reaction(  "OH + OH => H2O + O",  [5.72000E+12, 0, 390])

#  Reaction 19
# 19
#
reaction(  "OH + CO => CO2 + H",  [1.57000E+07, 1.3, -385])

#  Reaction 20
# 20
#  M'
three_body_reaction( "O + CO + M => CO2 + M",  [5.40000E+15, 0, 2300])

#  Reaction 21
# 21
#  M'
three_body_reaction( "H + CO + M => CHO + M",  [5.00000E+14, 0, 755])

#  Reaction 22
# 22
#
```

```
reaction(  "CH4 + O => CH3 + OH",  [4.07000E+14, 0, 7040])

#  Reaction 23
# 23
#
reaction(  "CH4 + H => CH3 + H2",  [7.24000E+14, 0, 7590])

#  Reaction 24
# 24
#
reaction(  "CH4 + OH => CH3 + H2O",  [1.55000E+06, 2.13, 1230])

#  Reaction 25
# 25
#
three_body_reaction( "CH4 + M => CH3 + H + M",  [4.68000E+17, 0,
46910])

#  Reaction 26
# 26
#
reaction(  "CH3 + O => CH2O + H",  [6.02000E+13, 0, 0])

#  Reaction 27
# 27
#
reaction(  "CH2O + O => CHO + OH",  [1.82000E+13, 0, 1550])

#  Reaction 28
# 28
#
reaction(  "CH2O + H => CHO + H2",  [3.31000E+14, 0, 5290])

#  Reaction 29
# 29
#
reaction(  "CH2O + OH => CHO + H2O",  [7.58000E+12, 0, 72])

#  Reaction 30
# 30
#
reaction(  "CHO + O2 => CO + HO2",  [3.00000E+12, 0, 0])

#  Reaction 31
# 31
#
reaction(  "CHO + H => CO + H2",  [4.00000E+13, 0, 0])

#  Reaction 32
# 32
#
reaction(  "CHO + OH => CO + H2O",  [5.00000E+12, 0, 0])

#  Reaction 33
# 33
#
reaction(  "CHO + O => CO + OH",  [1.00000E+13, 0, 0])
```

```
#  Reaction 34
# 34
#
reaction(  "CH2O + CH3 => CHO + CH4",  [2.23000E+13, 0, 2590])

#  Reaction 35
# 35
#
reaction(  "CH3 + OH => CH2O + H2",  [3.98000E+12, 0, 0])

#  Reaction 36
# 36
#
reaction(  "CH3 + HO2 => CH4 + O2",  [1.02000E+12, 0, 200])

#  Reaction 37
# 37
#
reaction(  "CO + HO2 => CO2 + OH",  [1.50000E+14, 0, 11900])

#  Reaction 38
# 38
#
reaction(  "CH3 + CH3 => C2H6",  [4.56000E+17, -7.65, 4250])

#  Reaction 39
# 39
#
reaction(  "C2H6 + O => C2H5 + OH",  [2.51000E+13, 0, 3200])

#  Reaction 40
# 40
#
reaction(  "C2H6 + H => C2H5 + H2",  [5.00000E+02, 3.5, 2620])

#  Reaction 41
# 41
#
reaction(  "C2H6 + OH => C2H5 + H2O",  [6.63000E+13, 0, 675])

#  Reaction 42
# 42
#
reaction(  "C2H5 + H => C2H6",  [7.23000E+13, 0, 0])

#  Reaction 43
# 43
#
reaction(  "C2H5 + H => CH3 + CH3",  [3.73000E+13, 0, 0])

#  Reaction 44
# 44
#
reaction(  "C2H5 => C2H4 + H",  [2.29000E+11, 0, 19120])

#  Reaction 45
```

```
# 45
#
reaction(  "C2H5 + O2 => C2H4 + HO2",  [1.53000E+12, 0, 2446])

#  Reaction 46
# 46
#
reaction(  "C2H4 + O => CH2 + CH2O",  [2.53000E+13, 0, 2516])

#  Reaction 47
# 47
#
reaction(  "C2H4 + OH => CH2O + CH3",  [5.00000E+13, 0, 3020])

#  Reaction 48
# 48
#
reaction(  "C2H4 + O => C2H3 + OH",  [2.53000E+13, 0, 2516])

#  Reaction 49
# 49
#
reaction(  "C2H4 + O2 => C2H3 + HO2",  [1.33000E+15, 0, 27680])

#  Reaction 50
# 50
#
reaction(  "C2H4 + H => C2H3 + H2",  [2.00000E+15, 0, 10000])

#  Reaction 51
# 51
#
reaction(  "C2H4 + OH => C2H3 + H2O",  [4.40000E+14, 0, 3270])

#  Reaction 52
# 52
#
three_body_reaction( "C2H3 + M => C2H2 + H + M",  [3.01000E+16, 0,
20380])

#  Reaction 53
# 53
#
reaction(  "C2H3 + O2 => C2H2 + HO2",  [1.57000E+13, 0, 5030])

#  Reaction 54
# 54
#
reaction(  "C2H3 + H => C2H2 + H2",  [7.53000E+13, 0, 0])

#  Reaction 55
# 55
#
reaction(  "C2H3 + OH => C2H2 + H2O",  [1.00000E+13, 0, 0])

#  Reaction 56
# 56
```

```
#
reaction(  "C2H2 + OH => CH3 + CO",  [5.48000E+13, 0, 6890])

#  Reaction 57
# 57
#
reaction(  "CH3 + H => CH2 + H2",  [2.00000E+11, 0.70, -1500])

#  Reaction 58
# 58
#
reaction(  "CH3 + OH => CH2 + H2O",  [6.00000E+10, 0.70, 1010])

#  Reaction 59
# 59
#
reaction(  "CH2 + O2 => CHO + OH",  [1.00000E+14, 0, 1860])

#  Reaction 60
# 60
#
reaction(  "CH2 + O2 => CH2O + O",  [1.00000E+14, 0, 1860])

#  Reaction 61
# 61
#
reaction(  "CH2 + O2 => CO2 + H2",  [1.00000E+14, 0, 1860])

#  Reaction 62
# 62
#
reaction(  "CH2 + H => CH + H2",  [4.00000E+13, 0, 0])

#  Reaction 63
# 63
#
reaction(  "CH + O => CO + H",  [4.00000E+13, 0, 0])

#  Reaction 64
# 64
#
reaction(  "CH + O2 => CO + OH",  [2.00000E+13, 0, 0])

#  Reaction 65
# 65
#
reaction(  "C2H + O => CO + CH",  [1.00000E+13, 0, 0])

#  Reaction 66
# 66
#
three_body_reaction( "CH* + M => CH + M",  [4.00000E+10, 0.50, 0])

#  Reaction 67
# 67
#
reaction(  "CH* + O2 => CH + O2",  [2.40000E+12, 0.50, 0])
```

```
#  Reaction 68
# 68
#
reaction(  "CH* => CH",  [1.90000E+06, 0, 0])

#  Reaction 69
# 69
#
reaction(  "C2H + O2 => CH* + CO2",  [4.50000E+15, 0, 25000])

#  Reaction 70
# 70
#
reaction(  "C2H + O => CH* + CO",  [7.10000E+11, 0, 0])

#  Reaction 71
# 71
#
reaction(  "C2H2 + H => C2H + H2",  [6.00000E+13, 0, 23650])

#  Reaction 72
# 72
#
reaction(  "C2H2 + OH => C2H + H2O",  [1.00000E+13, 0, 7000])

#  Reaction 73
# 73
#
reaction(  "C2H + O2 => CO + CHO",  [5.00000E+13, 0, 1505])

#  Reaction 74
# 74
#
reaction(  "CH + O => CHO+ + E",  [2.52000E+11, 0, 1700])

#  Reaction 75
# 75
#
reaction(  "CH* + O => CHO+ + E",  [5.04000E+14, 0, 1700])

#  Reaction 76
# 76
#
reaction(  "CHO+ + H2O => H3O+ + CO",  [1.00000E+16, -0.0897, 0])

#  Reaction 77
# 77
#
# reaction(  "H3O+ + C2H2 => C2H3O+ + H2",  [8.39000E+15, 0, 0])

#  Reaction 78
# 78
#
reaction(  "CHO+ + CH2 => CH3+ + CO",  [5.62000E+14, -0.006000, 0])

#  Reaction 79
```

```
# 79
#
reaction(  "H3O+ + CH2 => CH3+ + H2O",  [6.17000E+14, -0.006000, 0])

#  Reaction 80
# 80
#
reaction(  "CH3+ + C2H2 => C3H3+ + H2",  [7.24000E+14, 0, 0])

#  Reaction 81
# 81
#
# reaction(  "C3H3+ + H2O => C2H3O+ + CH2",  [7.24000E+14, 0, 0])

#  Reaction 82
# 82
#
# reaction(  "CH3+ + CO2 => C2H3O+ + O",  [7.24000E+14, 0, 0])

#  Reaction 83
# 83
#
reaction(  "H3O+ + E => H2O + H",  [2.29000E+18, -0.50, 0])

#  Reaction 84
# 84
# For now (instead of just Products)
reaction(  "C3H3+ + E => C2H2 + CH",  [1.50000E+19, -0.50, 0])

#  Reaction 85
# 85
#
reaction(  "CH3+ + E => CH2 + H",  [2.29000E+18, -0.50, 0])
```

B-4: Listing of Peters Mechanism

```
#
# Generated from file Peters.che
# by ck2cti on Wed Aug  3 16:32:16 2005
#
# Transport data from file PetersTrans.dat.

units(length = "cm", time = "s", quantity = "mol", act_energy =
"kJ/mol")


ideal_gas(name = "gas",
      elements = " C  H  O  N  E ",
      species = """ CH4  C2H  C2H2  C2H3  C2H4  C2H5  C2H6  C3H3  C3H4
C3H5
                    C3H6  C3H8  CH  CH2  CH2O  CH3  CHCO  CHO  CO  CO2
                    H  H2  H2O  H2O2  HO2  O  O2  OH  i-C3H7  n-C3H7
                    E HCO+ H3O+ N2 """,
      reactions = "all",
      transport = "Mix",
      initial_state = state(temperature = 300.0,
                        pressure = OneAtm)    )



#-----------------------------------------------------------------------
---------
#  Species data
#-----------------------------------------------------------------------
---------

species(name = "CH4",
    atoms = " C:1  H:4 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  5.149876130E+00,  -1.367097880E-
02,
                4.918005990E-05,  -4.847430260E-08,   1.666939560E-11,
               -1.024664760E+04,  -4.641303760E+00] ),
       NASA( [ 1000.00,  3500.00], [  7.485149500E-02,   1.339094670E-
02,
               -5.732858090E-06,   1.222925350E-09,  -1.018152300E-13,
               -9.468344590E+03,   1.843731800E+01] )
             ),
    transport = gas_transport(
                      geom = "nonlinear",
                      diam =     3.75,
                      well_depth =   141.40,
                      polar =     2.60,
                      rot_relax =    13.00),
    note = "L 8/88"
       )

species(name = "C2H",
    atoms = " C:2  H:1 ",
```

```
    thermo = (
       NASA( [  200.00,  1000.00], [  2.889657330E+00,   1.340996110E-
02,
               -2.847695010E-05,   2.947910450E-08,  -1.093315110E-11,
                6.683939320E+04,   6.222964380E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.167806520E+00,   4.752219020E-
03,
               -1.837870770E-06,   3.041902520E-10,  -1.772327700E-14,
                6.712106500E+04,   6.635894750E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     4.10,
                     well_depth =   209.00,
                     rot_relax =     2.50),
    note = "L 1/91"
       )

species(name = "C2H2",
    atoms = " C:2  H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  8.086810940E-01,   2.336156290E-
02,
               -3.551718150E-05,   2.801524370E-08,  -8.500729740E-12,
                2.642898070E+04,   1.393970510E+01] ),
       NASA( [ 1000.00,  3500.00], [  4.147569640E+00,   5.961666640E-
03,
               -2.372948520E-06,   4.674121710E-10,  -3.612352130E-14,
                2.593599920E+04,  -1.230281210E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     4.10,
                     well_depth =   209.00,
                     rot_relax =     2.50),
    note = "L 1/91"
       )

species(name = "C2H3",
    atoms = " C:2  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.212466450E+00,   1.514791620E-
03,
                2.592094120E-05,  -3.576578470E-08,   1.471508730E-11,
                3.485984680E+04,   8.510540250E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.016724000E+00,   1.033022920E-
02,
               -4.680823490E-06,   1.017632880E-09,  -8.626070410E-14,
                3.461287390E+04,   7.787323780E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.10,
                     well_depth =   209.00,
                     rot_relax =     1.00),
    note = "L 2/92"
       )
```

```
species(name = "C2H4",
    atoms = " C:2  H:4 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.959201480E+00,  -7.570522470E-
03,
                5.709902920E-05,  -6.915887530E-08,   2.698843730E-11,
                5.089775930E+03,   4.097330960E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.036111160E+00,   1.464541510E-
02,
               -6.710779150E-06,   1.472229230E-09,  -1.257060610E-13,
                4.939886140E+03,   1.030536930E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.97,
                     well_depth =   280.80,
                     rot_relax =     1.50),
    note = "L 1/91"
       )

species(name = "C2H5",
    atoms = " C:2  H:5 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.306465680E+00,  -4.186588920E-
03,
                4.971428070E-05,  -5.991266060E-08,   2.305090040E-11,
                1.284162650E+04,   4.707209240E+00] ),
       NASA( [ 1000.00,  3500.00], [  1.954656420E+00,   1.739727220E-
02,
               -7.982066680E-06,   1.752176890E-09,  -1.496415760E-13,
                1.285752000E+04,   1.346243430E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.30,
                     well_depth =   252.30,
                     rot_relax =     1.50),
    note = "L12/92"
       )

species(name = "C2H6",
    atoms = " C:2  H:6 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.291424920E+00,  -5.501542700E-
03,
                5.994382880E-05,  -7.084662850E-08,   2.686857710E-11,
               -1.152220550E+04,   2.666823160E+00] ),
       NASA( [ 1000.00,  3500.00], [  1.071881500E+00,   2.168526770E-
02,
               -1.002560670E-05,   2.214120010E-09,  -1.900028900E-13,
               -1.142639320E+04,   1.511561070E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     4.30,
                     well_depth =   252.30,
```

```
                    rot_relax =     1.50),
    note = "L 8/88"
       )

species(name = "C3H3",
    atoms = " C:3  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  1.828407660E+00,   2.378390360E-
02,
               -2.192281760E-05,   1.000674440E-08,  -1.389846440E-12,
                4.018630580E+04,   1.384479570E+01] ),
       NASA( [ 1000.00,  6000.00], [  6.641758210E+00,   8.085874280E-
03,
               -2.847878870E-06,   4.535259770E-10,  -2.688798150E-14,
                3.897936990E+04,  -1.040042550E+01] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     4.76,
                    well_depth =   252.00,
                    rot_relax =     1.00),
    note = "BUR 92"
       )

species(name = "C3H4",
    atoms = " C:3  H:4 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.613074870E+00,   1.212233710E-
02,
                1.854054000E-05,  -3.452584750E-08,   1.533533890E-11,
                2.154156420E+04,   1.025033190E+01] ),
       NASA( [ 1000.00,  6000.00], [  6.316948690E+00,   1.113362620E-
02,
               -3.962890180E-06,   6.356337750E-10,  -3.787498850E-14,
                2.011746170E+04,  -1.097188620E+01] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     4.76,
                    well_depth =   252.00,
                    rot_relax =     1.00),
    note = "L12/92"
       )

species(name = "C3H5",
    atoms = " C:3  H:5 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.787946930E+00,   9.484143350E-
03,
                2.423433680E-05,  -3.656040100E-08,   1.485923560E-11,
                1.862612180E+04,   7.828224990E+00] ),
       NASA( [ 1000.00,  6000.00], [  6.547611320E+00,   1.331522460E-
02,
               -4.783331000E-06,   7.719498140E-10,  -4.619308080E-14,
                1.727147070E+04,  -9.274868410E+00] )
             ),
    transport = gas_transport(
```

```
                    geom = "nonlinear",
                    diam =     4.98,
                    well_depth =   266.80,
                    rot_relax =     1.00),
    note = "BUR 92"
       )

species(name = "C3H6",
    atoms = " C:3  H:6 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.834645240E+00,   3.290784050E-
03,
                5.052281840E-05,  -6.662514180E-08,   2.637075850E-11,
                7.538382950E+02,   7.534109950E+00] ),
       NASA( [ 1000.00,  6000.00], [  6.038704990E+00,   1.629638950E-
02,
               -5.821306240E-06,   9.359364830E-10,  -5.586029030E-14,
               -7.765950920E+02,  -8.438243220E+00] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     4.98,
                    well_depth =   266.80,
                    rot_relax =     1.00),
    note = "L 7/90"
       )

species(name = "C3H8",
    atoms = " C:3  H:8 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  9.335538100E-01,   2.642457900E-
02,
                6.105972700E-06,  -2.197749900E-08,   9.514925300E-12,
               -1.395852000E+04,   1.920169100E+01] ),
       NASA( [ 1000.00,  5000.00], [  7.534136800E+00,   1.887223900E-
02,
               -6.271849100E-06,   9.147564900E-10,  -4.783806900E-14,
               -1.646751600E+04,  -1.789234900E+01] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     4.98,
                    well_depth =   266.80,
                    rot_relax =     1.00),
    note = "L 4/85"
       )

species(name = "CH",
    atoms = " C:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.489816650E+00,   3.238355410E-
04,
               -1.688990650E-06,   3.162173270E-09,  -1.406090670E-12,
                7.079729340E+04,   2.084011080E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.878464730E+00,   9.709136810E-
04,
                1.444456550E-07,  -1.306878490E-10,   1.760793830E-14,
```

```
                7.101243640E+04,   5.484979990E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "TPIS79"
       )

species(name = "CH2",
    atoms = " C:1  H:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.762678670E+00,   9.688721430E-
04,
                2.794898410E-06,  -3.850911530E-09,   1.687417190E-12,
                4.600404010E+04,   1.562531850E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.874101130E+00,   3.656392920E-
03,
               -1.408945970E-06,   2.601795490E-10,  -1.877275670E-14,
                4.626360400E+04,   6.171193240E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.80,
                     well_depth =   144.00),
    note = "L S/93"
       )

species(name = "CH2O",
    atoms = " H:2  C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.793723150E+00,  -9.908333690E-
03,
                3.732200080E-05,  -3.792852610E-08,   1.317726520E-11,
               -1.430895670E+04,   6.028129000E-01] ),
       NASA( [ 1000.00,  3500.00], [  1.760690080E+00,   9.200000820E-
03,
               -4.422588130E-06,   1.006412120E-09,  -8.838556400E-14,
               -1.399583230E+04,   1.365632300E+01] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00,
                     rot_relax =     2.00),
    note = "L 8/88"
       )

species(name = "CH3",
    atoms = " C:1  H:3 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.673590400E+00,   2.010951750E-
03,
                5.730218560E-06,  -6.871174250E-09,   2.543857340E-12,
                1.644499880E+04,   1.604564330E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.285717720E+00,   7.239900370E-
03,
```

```
               -2.987143480E-06,   5.956846440E-10,  -4.671543940E-14,
                1.677558430E+04,   8.480071790E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.80,
                     well_depth =   144.00),
    note = "L11/89"
       )

species(name = "CHCO",
    atoms = " H:1  C:2  O:1 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  2.251721400E+00,   1.765502100E-
02,
               -2.372910100E-05,   1.727575900E-08,  -5.066481100E-12,
                2.005944900E+04,   1.249041700E+01] ),
       NASA( [ 1000.00,  4000.00], [  5.628205800E+00,   4.085340100E-
03,
               -1.593454700E-06,   2.862605200E-10,  -1.940783200E-14,
                1.932721500E+04,  -3.930259500E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.50,
                     well_depth =   150.00,
                     rot_relax =     1.00),
    note = "SRIC91"
       )

species(name = "CHO",
    atoms = " H:1  C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.221185840E+00,  -3.243925320E-
03,
                1.377994460E-05,  -1.331440930E-08,   4.337688650E-12,
                3.839564960E+03,   3.394372430E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.772174380E+00,   4.956955260E-
03,
               -2.484456130E-06,   5.891617780E-10,  -5.335087110E-14,
                4.011918150E+03,   9.798344920E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.59,
                     well_depth =   498.00),
    note = "L12/89"
       )

species(name = "CO",
    atoms = " C:1  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.579533470E+00,  -6.103536800E-
04,
                1.016814330E-06,   9.070058840E-10,  -9.044244990E-13,
               -1.434408600E+04,   3.508409280E+00] ),
```

```
       NASA( [ 1000.00,  3500.00], [  2.715185610E+00,   2.062527430E-
03,
               -9.988257710E-07,   2.300530080E-10,  -2.036477160E-14,
               -1.415187240E+04,   7.818687720E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.65,
                    well_depth =    98.10,
                    polar =     1.95,
                    rot_relax =     1.80),
    note = "TPIS79"
       )

species(name = "CO2",
    atoms = " C:1  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.356773520E+00,   8.984596770E-
03,
               -7.123562690E-06,   2.459190220E-09,  -1.436995480E-13,
               -4.837196970E+04,   9.901052220E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.857460290E+00,   4.414370260E-
03,
               -2.214814040E-06,   5.234901880E-10,  -4.720841640E-14,
               -4.875916600E+04,   2.271638060E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.76,
                    well_depth =   244.00,
                    polar =     2.65,
                    rot_relax =     2.10),
    note = "L 7/88"
       )

species(name = "H",
    atoms = " H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  2.500000000E+00,   7.053328190E-
13,
               -1.995919640E-15,   2.300816320E-18,  -9.277323320E-22,
                2.547365990E+04,  -4.466828530E-01] ),
       NASA( [ 1000.00,  3500.00], [  2.500000010E+00,  -2.308429730E-
11,
                1.615619480E-14,  -4.735152350E-18,   4.981973570E-22,
                2.547365990E+04,  -4.466829140E-01] )
             ),
    transport = gas_transport(
                    geom = "atom",
                    diam =     2.05,
                    well_depth =   145.00),
    note = "L 7/88"
       )

species(name = "H2",
    atoms = " H:2 ",
    thermo = (
```

```
       NASA( [  200.00,  1000.00], [  2.344331120E+00,   7.980520750E-
03,
               -1.947815100E-05,   2.015720940E-08,  -7.376117610E-12,
               -9.179351730E+02,   6.830102380E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.337279200E+00,  -4.940247310E-
05,
                4.994567780E-07,  -1.795663940E-10,   2.002553760E-14,
               -9.501589220E+02,  -3.205023310E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     2.92,
                     well_depth =    38.00,
                     polar =     0.79,
                     rot_relax =   280.00),
    note = "TPIS78"
       )

species(name = "H2O",
    atoms = " H:2  O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.198640560E+00,  -2.036434100E-
03,
                6.520402110E-06,  -5.487970620E-09,   1.771978170E-12,
               -3.029372670E+04,  -8.490322080E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.033992490E+00,   2.176918040E-
03,
               -1.640725180E-07,  -9.704198700E-11,   1.682009920E-14,
               -3.000429710E+04,   4.966770100E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     2.60,
                     well_depth =   572.40,
                     dipole =     1.84,
                     rot_relax =     4.00),
    note = "L 8/89"
       )

species(name = "H2O2",
    atoms = " H:2  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.276112690E+00,  -5.428224170E-
04,
                1.673357010E-05,  -2.157708130E-08,   8.624543630E-12,
               -1.770258210E+04,   3.435050740E+00] ),
       NASA( [ 1000.00,  3500.00], [  4.165002850E+00,   4.908316940E-
03,
               -1.901392250E-06,   3.711859860E-10,  -2.879083050E-14,
               -1.786178770E+04,   2.916156620E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.46,
                     well_depth =   107.40,
                     rot_relax =     3.80),
    note = "L 7/88"
```

```
       )

species(name = "HO2",
    atoms = " H:1  O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.301798010E+00,  -4.749120510E-
03,
                2.115828910E-05,  -2.427638940E-08,   9.292251240E-12,
                2.948080400E+02,   3.716662450E+00] ),
       NASA( [ 1000.00,  3500.00], [  4.017210900E+00,   2.239820130E-
03,
               -6.336581500E-07,   1.142463700E-10,  -1.079085350E-14,
                1.118567130E+02,   3.785102150E+00] )
             ),
    transport = gas_transport(
                     geom = "nonlinear",
                     diam =     3.46,
                     well_depth =   107.40,
                     rot_relax =     1.00),
    note = "L 5/89"
       )

species(name = "O",
    atoms = " O:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.168267100E+00,  -3.279318840E-
03,
                6.643063960E-06,  -6.128066240E-09,   2.112659710E-12,
                2.912225920E+04,   2.051933460E+00] ),
       NASA( [ 1000.00,  3500.00], [  2.569420780E+00,  -8.597411370E-
05,
                4.194845890E-08,  -1.001777990E-11,   1.228336910E-15,
                2.921757910E+04,   4.784338640E+00] )
             ),
    transport = gas_transport(
                     geom = "atom",
                     diam =     2.75,
                     well_depth =    80.00),
    note = "L 1/90"
       )

species(name = "O2",
    atoms = " O:2 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.782456360E+00,  -2.996734160E-
03,
                9.847302010E-06,  -9.681295090E-09,   3.243728370E-12,
               -1.063943560E+03,   3.657675730E+00] ),
       NASA( [ 1000.00,  3500.00], [  3.282537840E+00,   1.483087540E-
03,
               -7.579666690E-07,   2.094705550E-10,  -2.167177940E-14,
               -1.088457720E+03,   5.453231290E+00] )
             ),
    transport = gas_transport(
                     geom = "linear",
                     diam =     3.46,
                     well_depth =   107.40,
```

```
                    polar =     1.60,
                    rot_relax =     3.80),
    note = "TPIS89"
       )

species(name = "OH",
    atoms = " O:1  H:1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  3.992015430E+00,  -2.401317520E-
03,
                 4.617938410E-06,  -3.881133330E-09,   1.364114700E-12,
                 3.615080560E+03,  -1.039254580E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.092887670E+00,   5.484297160E-
04,
                 1.265052280E-07,  -8.794615560E-11,   1.174123760E-14,
                 3.858657000E+03,   4.476696100E+00] )
             ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     2.75,
                    well_depth =    80.00),
    note = "RUS 78"
       )

species(name = "i-C3H7",
    atoms = " C:3  H:7 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  5.408728720E+00,  -8.552218250E-
03,
                 8.421784910E-05,  -1.009426830E-07,   3.869144790E-11,
                 9.426009560E+03,   3.623225040E+00] ),
       NASA( [ 1000.00,  6000.00], [  5.751258820E+00,   1.876057620E-
02,
                -6.701919760E-06,   1.077518710E-09,  -6.430908850E-14,
                 7.979772930E+03,  -4.913593550E+00] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     4.98,
                    well_depth =   266.80,
                    rot_relax =     1.00),
    note = "L 9/85"
       )

species(name = "n-C3H7",
    atoms = " C:3  H:7 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.032399960E+00,   3.427283120E-
03,
                 6.143444200E-05,  -8.376463380E-08,   3.408577760E-11,
                 1.033938390E+04,   8.774280790E+00] ),
       NASA( [ 1000.00,  6000.00], [  6.964684620E+00,   1.754519460E-
02,
                -6.233700550E-06,   9.985297350E-10,  -5.943947930E-14,
                 8.542443580E+03,  -1.148314780E+01] )
             ),
    transport = gas_transport(
```

```
                    geom = "nonlinear",
                    diam =     4.98,
                    well_depth =   266.80,
                    rot_relax =     1.00),
    note = "L 6/90"
       )

species(name = "E",
    atoms = " E:1 ",
    thermo = (
       NASA( [  200.00,     5.49], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
               -7.453750000E+02,  -1.172469020E+01] ),
       NASA( [    5.49,  6000.00], [  2.500000000E+00,
0.000000000E+00,
                0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
               -7.453750000E+02,  -1.172469020E+01] )
             ),
    transport = gas_transport(
                    geom = "atom",
                    diam =   425.00,
                    well_depth =   850.00,
                    rot_relax =     1.00),
    note = "L10/92"
       )

species(name = "HCO+",
    atoms = " H:1  C:1  O:1  E:-1 ",
    thermo = (
       NASA( [  300.00,    29.02], [  2.473973600E+00,   8.671559000E-
03,
               -1.003150000E-05,   6.717052700E-09,  -1.787267400E-12,
                9.914660800E+04,   8.175711870E+00] ),
       NASA( [   29.02,  5000.00], [  3.741188000E+00,   3.344151700E-
03,
               -1.239712100E-06,   2.118938800E-10,  -1.370415000E-14,
                9.888407800E+04,   2.078613570E+00] )
             ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     3.59,
                    well_depth =   498.00),
    note = "J12/70"
       )

species(name = "H3O+",
    atoms = " H:3  O:1  E:-1 ",
    thermo = (
       NASA( [  200.00,  1000.00], [  4.198640560E+00,  -2.036434100E-
03,
                6.520402110E-06,  -5.487970620E-09,   1.771978170E-12,
               -3.029372670E+04,  -8.490322080E-01] ),
       NASA( [ 1000.00,  3500.00], [  3.033992490E+00,   2.176918040E-
03,
               -1.640725180E-07,  -9.704198700E-11,   1.682009920E-14,
               -3.000429710E+04,   4.966770100E+00] )
```

```
            ),
    transport = gas_transport(
                    geom = "nonlinear",
                    diam =     2.60,
                    well_depth =   572.40,
                    dipole =     1.84,
                    rot_relax =     4.00),
    note = "L 8/89"
       )

species(name = "N2",
    atoms = " N:2 ",
    thermo = (
       NASA( [  300.00,  1000.00], [  3.298677000E+00,   1.408240400E-
03,
               -3.963222000E-06,   5.641515000E-09,  -2.444854000E-12,
               -1.020899900E+03,   3.950372000E+00] ),
       NASA( [ 1000.00,  5000.00], [  2.926640000E+00,   1.487976800E-
03,
               -5.684760000E-07,   1.009703800E-10,  -6.753351000E-15,
               -9.227977000E+02,   5.980528000E+00] )
            ),
    transport = gas_transport(
                    geom = "linear",
                    diam =     3.62,
                    well_depth =    97.53,
                    polar =     1.76,
                    rot_relax =     4.00),
    note = "121286"
       )



#-----------------------------------------------------------------------
---------
#  Reaction data
#-----------------------------------------------------------------------
---------

#1.1
#H2/O2 Chain Reactions
reaction("O2 + H => OH + O",  [2.000E+14, 0.00, 70.30])  #1f
reaction("OH + O => O2 + H",  [1.568E+13, 0.00,  3.52])  #1b
reaction("H2 + O => OH + H",  [5.060E+04, 2.67, 26.30])  #2f
reaction("OH + H => H2 + O",  [2.222E+04, 2.67, 18.29])  #2b
reaction("H2 + OH => H2O + H", [1.000E+08, 1.60, 13.80]) #3f
reaction("H2O + H => H2 + OH", [4.312E+08, 1.60, 76.46]) #3b
reaction("OH + OH => H2O + O", [1.500E+09, 1.14,  0.42]) #4f
reaction("H2O + O => OH + OH", [1.473E+10, 1.14, 71.09]) #4b

#1.2
#HO2 Formation and Consumption
three_body_reaction("O2 + H + M => HO2 + M", [2.300E+18, -0.80, 0.00],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #5f
three_body_reaction("HO2 + M => O2 + H + M", [3.190E+18, -0.80,
195.39],
```

```
                efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75 O2:0.4
N2:0.4 ") #5b
reaction("HO2 + H => OH + OH", [1.500E+14, 0.00, 4.20])  #6
reaction("HO2 + H => H2 + O2", [2.500E+13, 0.00, 2.90])  #7
reaction("HO2 + OH => H2O + O2", [6.000E+13, 0.00, 0.00])  #8
reaction("HO2 + H => H2O + O", [3.000E+13, 0.00, 7.20])  #9
reaction("HO2 + O => OH + O2", [1.800E+13, 0.00, -1.70])  #10

#1.3
#H2O2 Formation and Consumption
reaction("HO2 + HO2 => H2O2 + O2", [2.500E+11, 0.00, -5.20])  #11
three_body_reaction("OH + OH + M => H2O2 + M", [3.250E+22, -2.00,
0.00],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #12f
three_body_reaction("H2O2 + M => OH + OH + M", [1.692E+24, -2.00,
202.29],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #12b
reaction("H2O2 + H => H2O + OH", [1.000E+13, 0.00, 15.00]) #13
reaction("H2O2 + H => H2 + HO2", [1.700E+12, 0.00, 15.70]) #14f
reaction("H2 + HO2 => H2O2 + H", [1.150E+12, 0.00, 80.88]) #14b

#1.4
#Recombination Reactions
three_body_reaction("H + H + M => H2 + M", [1.800E+18, -1.00, 0.00],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #15
three_body_reaction("OH + H + M => H2O + M", [2.200E+22, -2.00, 0.00],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #16
three_body_reaction("O + O + M => O2 + M", [2.900E+17, -1.00, 0.00],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #17

#2.
#CO/CO2 Mechanism
reaction("CO + OH => CO2 + H", [4.400E+06, 1.50, -3.10]) #18f
reaction("CO2 + H => CO + OH", [4.956E+08, 1.50, 89.76]) #18b

#3.1
#CH Consumption
reaction("CH + O2 => CHO + O", [3.000E+13, 0.00, 0.00]) #19
reaction("CO2 + CH => CHO + CO", [3.400E+12, 0.00, 2.90]) #20

#3.2
#CHO Consumption
reaction("CHO + H => CO + H2", [2.000E+14, 0.00, 0.00]) #21
reaction("CHO + OH => CO + H2O", [1.000E+14, 0.00, 0.00]) #22
reaction("CHO + O2 => CO + HO2", [3.000E+12, 0.00, 0.00]) #23
three_body_reaction("CHO + M => CO + H + M", [7.100E+14, 0.00, 70.30],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #24f
three_body_reaction("CO + H + M => CHO + M", [1.136E+15, 0.00, 9.97],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #24b
```

```
#3.3
#CH2 Consumption
reaction("CH2 + H => CH + H2", [8.400E+09, 1.50, 1.40]) #25f
reaction("CH + H2 => CH2 + H", [5.830E+09, 1.50, 13.08]) #25b
reaction("CH2 + O => CO + H + H", [8.000E+13, 0.00, 0.00]) #26
reaction("CH2 + O2 => CO + OH + H", [6.500E+12, 0.00, 6.30]) #27
reaction("CH2 + O2 => CO2 + H + H", [6.500E+12, 0.00, 6.30]) #28

#3.4
#CH2O Consumption
reaction("CH2O + H => CHO + H2", [2.500E+13, 0.00, 16.70]) #29
reaction("CH2O + O => CHO + OH", [3.500E+13, 0.00, 14.60]) #30
reaction("CH2O + OH => CHO + H2O", [3.000E+13, 0.00, 5.00]) #31
three_body_reaction("CH2O + M => CHO + H + M", [1.400E+17, 0.00,
320.00],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #32

#3.5
#CH3 Consumption
reaction("CH3 + H => CH2 + H2", [1.800E+14, 0.00, 63.00]) #33f
reaction("CH2 + H2 => CH3 + H", [3.680E+13, 0.00, 44.30]) #33b
falloff_reaction("CH3 + H (+ M) => CH4 (+ M)",
                 kf = [2.108E+14, 0.00, 0.00],
              kf0 = [6.257E+23, -1.80, 0.00]) #34
reaction("CH3 + O => CH2O + H", [7.000E+13, 0.00, 0.00]) #35
falloff_reaction("CH3 + CH3 (+ M) => C2H6 (+ M)",
                 kf = [3.613E+13, 0.00, 0.00],
              kf0 = [1.270E+41, -7.00, 11.56]) #36
reaction("CH3 + O2 => CH2O + OH", [3.400E+11, 0.00, 37.40]) #37
reaction("CH4 + H => CH3 + H2", [2.200E+04, 3.00, 36.60]) #38f
reaction("CH3 + H2 => CH4 + H", [8.391E+02, 3.00, 34.56]) #38b
reaction("CH4 + O => CH3 + OH", [1.200E+07, 2.10, 31.90]) #39
reaction("CH4 + OH => CH3 + H2O", [1.600E+06, 2.10, 10.30]) #40f
reaction("CH3 + H2O => CH4 + OH", [2.631E+05, 2.10, 70.92]) #40b

#4.1
#C2H Consumption
reaction("C2H + H2 => C2H2 + H", [1.100E+13, 0.00, 12.00]) #41f
reaction("C2H2 + H => C2H + H2", [5.270E+13, 0.00, 119.95]) #41b
reaction("C2H + O2 => CHCO + O", [5.000E+13, 0.00, 6.30]) #42

#4.2
#CHCO Consumption
reaction("CHCO + H => CH2 + CO", [3.000E+13, 0.00, 0.00]) #43f
reaction("CH2 + CO => CHCO + H", [2.361E+12, 0.00, -29.39]) #43b
reaction("CHCO + O => CO + CO + H", [1.000E+14, 0.00, 0.00]) #44

#4.3
#C2H2 Consumption
reaction("C2H2 + O => CH2 + CO", [4.100E+08, 1.50, 7.10]) #45
reaction("C2H2 + O => CHCO + H", [4.300E+14, 0.00, 50.70]) #46
reaction("C2H2 + OH => C2H + H2O", [1.000E+13, 0.00, 29.30]) #47f
reaction("C2H + H2O => C2H2 + OH", [9.000E+12, 0.00, -15.98]) #47b
reaction("C2H2 + CH => C3H3", [2.100E+14, 0.00, -0.50]) #48

#4.4
```

```
#C2H3 Consumption
reaction("C2H3 + H => C2H2 + H2", [3.000E+13, 0.00, 0.00]) #49
reaction("C2H3 + O2 => C2H2 + HO2", [5.400E+11, 0.00, 0.00]) #50
falloff_reaction("C2H3 (+ M) => C2H2 + H (+ M)",
                 kf = [2.000E+14, 0.00, 166.29],
             kf0 = [1.187E+42, -7.50, 190.40]) #51f
reaction("C2H2 + H => C2H3", [1.053E+14, 0.00, 3.39]) #51b

#4.5
#C2H4 Consumption
reaction("C2H4 + H => C2H3 + H2", [1.500E+14, 0.00, 42.70]) #52f
reaction("C2H3 + H2 => C2H4 + H", [9.605E+12, 0.00, 32.64]) #52b
reaction("C2H4 + O => CH3 + CO + H", [1.600E+09, 1.20, 3.10]) #53
reaction("C2H4 + OH => C2H3 + H2O", [3.000E+13, 0.00, 12.60]) #54f
reaction("C2H3 + H2O => C2H4 + OH", [8.283E+12, 0.00, 65.20]) #54b
three_body_reaction("C2H4 + M => C2H2 + H2 + M", [2.500E+17, 0.00,
319.80],
                    efficiencies = " CH4:6.5 H2O:6.5 CO2:1.5 CO:0.75
O2:0.4 N2:0.4 ") #55

#4.6
#C2H5 Consumption
reaction("C2H5 + H => CH3 + CH3", [3.000E+13, 0.00, 0.00]) #56f
reaction("CH3 + CH3 => C2H5 + H", [3.547E+12, 0.00, 49.68]) #56b
reaction("C2H5 + O2 => C2H4 + HO2", [2.000E+12, 0.00, 20.90]) #57
falloff_reaction("C2H5 (+ M) => C2H4 + H (+ M)",
                 kf = [2.000E+13, 0.00, 166.00],
             kf0 = [1.000E+17, 0.00, 130.00]) #58f
reaction("C2H4 + H => C2H5", [3.189E+13, 0.00, 12.61]) #58b

#4.7
#C2H6 Consumption
reaction("C2H6 + H => C2H5 + H2", [5.400E+02, 3.50, 21.80]) #59
reaction("C2H6 + O => C2H5 + OH", [3.000E+07, 2.00, 21.40]) #60
reaction("C2H6 + OH => C2H5 + H2O", [6.300E+06, 2.00, 2.70]) #61

#5.1
#C3H3 Consumption
reaction("C3H3 + O2 => CHCO + CH2O", [6.000E+12, 0.00, 0.00]) #62
reaction("C3H3 + O => C2H3 + CO", [3.800E+13, 0.00, 0.00]) #63
reaction("C3H4 => C3H3 + H", [5.000E+14, 0.00, 370.00]) #64f
reaction("C3H3 + H => C3H4", [1.700E+13, 0.00, 19.88]) #64b

#5.2
#C3H4 Consumption
reaction("C3H4 + O => C2H2 + CH2O", [1.000E+12, 0.00, 0.00]) #65
reaction("C3H4 + O => C2H3 + CHO", [1.000E+12, 0.00, 0.00]) #66
reaction("C3H4 + OH => C2H3 + CH2O", [1.000E+12, 0.00, 0.00]) #67
reaction("C3H4 + OH => C2H4 + CHO", [1.000E+12, 0.00, 0.00]) #68

#5.3
#C3H5 Consumption
reaction("C3H5 => C3H4 + H", [3.980E+13, 0.00, 293.10]) #69f
reaction("C3H4 + H => C3H5", [1.267E+13, 0.00, 32.48]) #69b
reaction("C3H5 + H => C3H4 + H2", [1.000E+13, 0.00, 0.00]) #70

#5.4
```

```
#C3H6 Consumption
reaction("C3H6 => C2H3 + CH3", [3.150E+15, 0.00, 359.30]) #71f
reaction("C2H3 + CH3 => C3H6", [2.511E+12, 0.00, -34.69]) #71b
reaction("C3H6 + H => C3H5 + H2", [5.000E+12, 0.00, 6.30]) #72

#5.5
#C3H7 Consumption
reaction("n-C3H7 => C2H4 + CH3", [9.600E+13, 0.00, 129.80]) #73
reaction("n-C3H7 => C3H6 + H", [1.250E+14, 0.00, 154.90]) #74f
reaction("C3H6 + H => n-C3H7", [4.609E+14, 0.00, 21.49]) #74b
reaction("i-C3H7 => C2H4 + CH3", [6.300E+13, 0.00, 154.50]) #75
reaction("i-C3H7 + O2 => C3H6 + HO2", [1.000E+12, 0.00, 20.90]) #76

#5.6
#C3H8 Consumption
reaction("C3H8 + H => n-C3H7 + H2", [1.300E+14, 0.00, 40.60]) #77
reaction("C3H8 + H => i-C3H7 + H2", [1.000E+14, 0.00, 34.90]) #78
reaction("C3H8 + O => n-C3H7 + OH", [3.000E+13, 0.00, 24.10]) #79
reaction("C3H8 + O => i-C3H7 + OH", [2.600E+13, 0.00, 18.70]) #80
reaction("C3H8 + OH => n-C3H7 + H2O", [3.700E+12, 0.00, 6.90]) #81
reaction("C3H8 + OH => i-C3H7 + H2O", [2.800E+12, 0.00, 3.60]) #82

#added from methane_ion31
#charged species reactions
reaction( "CH + O => HCO+ + E",   [5.75000E+11, 0, (6000, 'cal/mol')])
reaction( "HCO+ + H2O => CO + H3O+",   [5.02000E+17, 0, (24000,
'cal/mol')])
reaction( "H3O+ + E => H2O + H",   [1.44000E+17, 0, (0, 'cal/mol')])
```

Appendix C: Utility Codes

C-1: nameParse

Description:

The nameParse class is used to store essential information about a specific set of modeling results within a result file's name in a standardized format. This information includes the flow rates of each gas within the premixed flame in addition to the applied voltage across the modeled flame.

The nameParse class uses a separate filename format for methane and synthesis gas flames. In each filename format, labels are used prior to each stored attribute, and this label is separated by the corresponding value by an underscore. Examples of generated filenames are shown in the following figures for each type of fuel, and a table is given after each figure stating the meaning of each label.

V_0.0_mtot_0.00027051509604_mair_0.00025858_mfuel_1.193509604e-05.xml

Figure C-1: Example of Filename for a Methane Flame Model Output File

Label	Meaning
V	Applied Voltage Across the Modeled Flame Object
mtot	Total Mass Flux Through the Burner
mair	Mass Flux of Air Through the Burner
mfuel	Mass Flux of Methane Through the Burner

Table C-1: List of Filename Labels for Methane Flame Model Output Files

V_0.0_mtot_0.000230085693251_mair_0.000197160766326_mCO_2.35867863617e-05_mH2_1.3724605293e-06_mN2_7.5285255675e-06_mCH4_4.37154466667e-07.xml

Figure C-2: Example of Filename for a Synthesis Gas Flame Model Output File

Label	Meaning
V	Voltage Applied Across the Modeled Flame Object
mtot	Total Mass Flux Through the Burner
mair	Mass Flux of Air Through the Burner
mCO	Mass Flux of Carbon Monoxide Through the Burner
mH2	Mass Flux of Hydrogen Gas Through the Burner
mN2	Mass Flux of Nitrogen Gas Through the Burner Separate From Nitrogen in Air
mCH4	Mass Flux of Methane Through the Burner

Table C-2: List of Filename Labels for Synthesis Gas Flame Model Output Files

Code Overview:

The nameParse script contains a set of functions to generate a filename based on given flow rates and also to retrieve information about flow rates from an existing filename. In addition a set of "constants" are included within the script, but these values may be changed by the user. The following tables give explanation of constants and functions included within the nameParse code.

Name	Value	Description
N2_PERCENT	0.76708	Nitrogen gas percent by weight in air.
O2_PERCENT	0.23292	Oxygen percent by weight in air.
MW_O2	31.9998	Molecular weight of oxygen molecule
MW_N2	28.0134	Molecular weight of nitrogen molecule
MW_FUEL	16.13782	Molecular weight of methane based fuel
MW_CO	28.0101	Molecular weight of carbon monoxide
MW_H2	2.01588	Molecular weight of hydrogen gas
MW_CH4	16.04246	Molecular weight of methane

Table C-3: Table of Constant Values Included in the nameParse Class

Function	Purpose	Inputs	Outputs	Description
createNameMethane	Create a filename for a methane flame model output file	m_dot_total		Total mass flux at the burner.
		m_dot_air		Mass flux of air at the burner
		m_dot_fuel		Total mass flux of methane at the burner
		applied_voltage		Voltage applied across the model flame
			ret_str	Assembled filename for modeled flame output
createNameSyngas	Create a filename for a synthesis gas flame model output file	m_dot_total		Total mass flux at the burner.
		m_dot_air		Mass flux of air at the burner
		m_dot_co		Mass flux of carbon monoxide at the burner
		m_dot_h2		Mass flux of hydrogen gas at the burner
		m_dot_n2		Mass flux of nitrogen gas at the burner
		m_dot_ch4		Mass flux of methane at the burner
			ret_str	Assembled filename for modeled flame output
retrieveNameInfo	Used internally by the nameParse class to retrieve all information contained within an input filename originally created by the nameParse class	filename		The name of the file for which modeling properties are to be retrieved.
getAirFlux	Retrieves the air flux from a filename created using the nameParse class	filename		The name of the file from which to retrieve the air flux
			air_flux	The mass flux of air at the burner
getExtention	Retrieves the file extension for the input filename	filename		The name of the file from which to retrieve the file extension
			ext	The file extension included in the input file
getFuelFlux	Retrieves the methane flux at the burner for a methane flame model	filename		The name of the file from which to retrieve the methane fuel flux
			fuel_flux	The mass flux of methane fuel at the burner for methane flame models
getCOFlux	Retrieves the carbon monoxide	filename		The name of the file from which to retrieve

	flux at the burner for a synthesis gas flame model			the carbon monoxide flux
			co_flux	The mass flux of carbon monoxide at the burner in synthesis gas flame models
getH2Flux	Retrieves the hydrogen gas flux at the burner for a synthesis gas flame model	filename		The name of the file from which to retrieve the hydrogen gas flux
			h2_flux	The mass flux of hydrogen gas at the burner in synthesis gas flame models
getN2Flux	Retrieves the nitrogen gas flux not including nitrogen from air at the burner for a synthesis gas flame model	filename		The name of the file from which to retrieve the nitrogen gas flux
			n2_flux	The mass flux of nitrogen gas not included with air at the burner in synthesis gas flame models
getCH4Flux	Retrieves the methane flux at the burner for a synthesis gas flame model	filename		The name of the file from which to retrieve the methane flux
			ch4_flux	The mass flux of methane at the burner in synthesis gas flame models
getTotalFlux	Retrieves the total flux at the burner for a flame model	filename		The name of the file from which to retrieve the total flux
			total_flux	The total mass flux at the burner
getVoltage	Retrieves the voltage applied across a modeled flame	filename		The name of the file from which to retrieve the voltage
			voltage	The applied voltage across the modeled flame
getMoleComposition	Creates a formatted string containing the mole fractions of each species at the burner	filename		The name of the file from which to retrieve the molar composition
			ret_str	A formatted string containing the names of species input at the burner of the modeled flame and the corresponding mole fractions for each species
getMassComposition	Creates a formatted string containing the mass fractions of each species at the burner	filename		The name of the file from which to retrieve the mass composition
			ret_str	A formatted string containing the names of species input at the burner of the modeled

				flame and the corresponding mass fractions for each species
isValidFile	Determines if a given filename conforms to the format used by nameParse	filename		The name of the file to check for validity
			valid_file	Returns true if the input filename corresponds to the format used by nameParse, and false otherwise

Table C-4: Function Descriptions for the nameParse Class

Code Listing:

```
# Class used to create and interpret simulation file names for methane
and syngas mixes

class nameParse:
    """
    Used to form and parse information from filenames produced for
EFSim_NG_Current,
    EFSim_NG_Current_mass, EFSim_Syngas_Current, or
EFSim_Syngas_Current_mass classes.

    *** The Syngas classes should use the Carbon Monoxide, Hydrogen,
and Nitrogen inputs
    for the Syngas Fuel, but Methane (approximated natural gas) should
use the Fuel input.
    """
    #constants
    N2_PERCENT = 0.76708 #percent by mass N2 in air
    O2_PERCENT = 0.23292 #percent by mass O2 in air
    MW_O2 = 31.9998 #Oxygen molecular weight
    MW_N2 = 28.0134 #Nitrogen molecular weight
    MW_FUEL = 16.13782 #Fuel Molecular Weight
    MW_CO = 28.0101 #Carbon Monoxide Molecular Weight
    MW_H2 = 2.01588 #Hydrogen Molecular Weight
    MW_CH4 = 16.04246 #Methane Molecular Weight

    #variable
    air_flux = None
    ext = None
    fuel_flux = None
    total_flux = None
    valid_file = False
    voltage = None
    co_flux = None
    h2_flux = None
    n2_flux = None
    ch4_flux = None

    def createNameMethane(self, m_dot_total, m_dot_air, m_dot_fuel,
applied_voltage = None):
        """
        Return a string representing a filename without a '.' or
extention
```

```
        Inputs:
            m_dot_total = total mass flux
            m_dot_air = mass flux of air
            m_dot_fuel = mass flux of fuel in methane and air premix
            applied_voltage = applied voltage

        Outpus:
            ret_str = assembled filename string

        """
        ret_str = ''
        if applied_voltage != None:
            ret_str = ret_str + 'V_' + str(applied_voltage) + '_'
        ret_str = ret_str + 'mtot_' + str(m_dot_total) + '_'
        ret_str = ret_str + 'mair_' + str(m_dot_air) + '_'
        ret_str = ret_str + 'mfuel_' + str(m_dot_fuel)
        return ret_str

    def createNameSyngas(self, m_dot_total, m_dot_air, m_dot_co,
m_dot_h2, m_dot_n2, m_dot_ch4 = None, applied_voltage = None):
        """
        Return a string representing a filename without a '.' or
extention

        Inputs:
            m_dot_total = total mass flux
            m_dot_air = mass flux of air
            m_dot_co = mass flux of carbon monoxide in syngas and air
mix
            m_dot_h2 = mass flux of hydrogen gas in syngas air mix
            m_dot_n2 = mass flux of nitrogen gas from simulated syngas
in a syngas and air premix
            m_dot_ch4 = mass flux of methane in doped syngas mix
            applied_voltage = applied voltage

        Outpus:
            ret_str = assembled filename string

        """
        ret_str = ''
        if applied_voltage != None:
            ret_str = ret_str + 'V_' + str(applied_voltage) + '_'
        ret_str = ret_str + 'mtot_' + str(m_dot_total) + '_'
        ret_str = ret_str + 'mair_' + str(m_dot_air) + '_'
        ret_str = ret_str + 'mCO_' + str(m_dot_co) + '_'
        ret_str = ret_str + 'mH2_' + str(m_dot_h2) + '_'
        if m_dot_ch4 != None:
            ret_str = ret_str + 'mN2_' + str(m_dot_n2) + '_'
            ret_str = ret_str + 'mCH4_' + str(m_dot_ch4)
        else:
            ret_str = ret_str + 'mN2_' + str(m_dot_n2)
        return ret_str

    def retrieveNameInfo(self, filename):
        """
```

```
        Break a filename into voltage and flow information, if
possible.

        Input:
            filename = name of the file to check
        """
        #determine if file has extention
        if filename.find('.') == -1:
            self.valid_file = False
            self.voltage = None
            self.total_flux = None
            self.air_flux = None
            self.fuel_flux = None
            return
        #get extention
        position = -1
        while filename[position] != '.':
            position = position - 1;
        self.ext = filename[(position + 1):]

        #parse
        name = filename[:position]
        name = name.split('_')

        #interpret
        self.valid_file = False
        self.voltage = None
        self.total_flux = None
        self.air_flux = None
        self.fuel_flux = None
        self.co_flux = None
        self.h2_flux = None
        self.n2_flux = None

        for i in range (0, len(name) - 1):
            if name[i] == 'V':
                try:
                    self.voltage = float(name[i+1])
                except:
                    self.voltage = None
            elif name[i] == 'mtot':
                try:
                    self.total_flux = float(name[i+1])
                except:
                    self.total_flux = None
            elif name[i] == 'mair':
                try:
                    self.air_flux = float(name[i+1])
                except:
                    self.air_flux = None
            elif name[i] == 'mfuel':
                try:
                    self.fuel_flux = float(name[i+1])
                except:
                    self.fuel_flux = None
            elif name[i] == 'mCO':
                try:
```

```
                    self.co_flux = float(name[i+1])
                except:
                    self.co_flux = None
            elif name[i] == 'mH2':
                try:
                    self.h2_flux = float(name[i+1])
                except:
                    self.h2_flux = None
            elif name[i] == 'mN2':
                try:
                    self.n2_flux = float(name[i+1])
                except:
                    self.n2_flux = None
            elif name[i] == 'mCH4':
                try:
                    self.ch4_flux = float(name[i+1])
                except:
                    self.ch4_flux = None

        if self.total_flux == None or self.air_flux == None or
(self.fuel_flux == None and (self.co_flux == None or self.h2_flux ==
None or self.n2_flux == None)):
            self.valid_file = False
        else:
            self.valid_file = True

    def getAirFlux(self, filename):
        """
        Gets the Air Flux from a filename.

        Input: filename
        Output: air flux
        """
        self.retrieveNameInfo(filename)
        return self.air_flux

    def getExtention(self, filename):
        """
        Gets the Extention from a filename.

        Input: filename
        Output: file extention
        """
        self.retrieveNameInfo(filename)
        return self.ext

    def getFuelFlux(self, filename):
        """
        Gets the Fuel Flux from a filename.

        Input: filename
        Output: fuel flux
        """
        self.retrieveNameInfo(filename)
        return self.fuel_flux

    def getCOFlux(self, filename):
```

```
        """
        Gets the CO Flux from a filename.

        Input: filename
        Output: CO flux
        """
        self.retrieveNameInfo(filename)
        return self.co_flux

    def getH2Flux(self, filename):
        """
        Gets the H2 Flux from a filename.

        Input: filename
        Output: H2 flux
        """
        self.retrieveNameInfo(filename)
        return self.h2_flux

    def getN2Flux(self, filename):
        """
        Gets the N2 Flux from a filename.

        Input: filename
        Output: N2 flux
        """
        self.retrieveNameInfo(filename)
        return self.n2_flux

    def getCH4Flux(self, filename):
        """
        Gets the CH4 Flux from a filename.

        Input: filename
        Output: CH4 flux
        """
        self.retrieveNameInfo(filename)
        return self.ch4_flux

    def getTotalFlux(self, filename):
        """
        Gets the Total Flux from a filename.

        Input: filename
        Output: total mass flux
        """
        self.retrieveNameInfo(filename)
        return self.total_flux


    def getVoltage(self, filename):
        """
        Gets the Voltage from a filename.

        Input: filename
        Output: applied Voltage
        """
```

```
        self.retrieveNameInfo(filename)
        return self.voltage

    def getMoleComposition(self, filename):
        """
        Retrieves a Cantera format string for molar concentrations
based on an existing filename

        Input: filename
        Output: molar concentration string
        """
        self.retrieveNameInfo(filename)
        if self.valid_file == False:
            print "Not a valid filename!"
            return
        if self.fuel_flux != None: #is this methane fuel only
            o2_mole = (self.air_flux*self.O2_PERCENT)/self.MW_O2
            n2_mole = (self.air_flux*self.N2_PERCENT)/self.MW_N2
            ch4_mole = self.fuel_flux/self.MW_FUEL
            return 'O2:' + str(o2_mole) + ', N2:' + str(n2_mole) + ',
CH4:' + str(ch4_mole)
        else:
            o2_mole = (self.air_flux*self.O2_PERCENT)/self.MW_O2
            n2_mole =
(self.air_flux*self.N2_PERCENT+self.n2_flux)/self.MW_N2
            h2_mole = self.h2_flux/self.MW_H2
            co_mole = self.co_flux/self.MW_CO
            ch4_mole = self.ch4_flux/self.MW_CH4
            return 'O2:' + str(o2_mole) + ', N2:' + str(n2_mole) + ',
CH4:' + str(ch4_mole)+ ', H2:' + str(h2_mole)+ ', CO:' + str(co_mole)

    def getMassComposition(self, filename):
        """
        Retrieves a Cantera format string for mass concentrations based
on an existing filename

        Input: filename
        Output: mass concentration string
        """
        self.retrieveNameInfo(filename)
        if self.valid_file == False:
            print "Not a valid filename!"
            return
        if self.fuel_flux != None: #is this methane fuel only
            o2_mass = (self.air_flux*self.O2_PERCENT)
            n2_mass = (self.air_flux*self.N2_PERCENT)
            ch4_mass = self.fuel_flux
            return 'O2:' + str(o2_mass) + ', N2:' + str(n2_mass) + ',
CH4:' + str(ch4_mass)
        else:
            o2_mass = (self.air_flux*self.O2_PERCENT)
            n2_mass = (self.air_flux*self.N2_PERCENT+self.n2_flux)
            h2_mass = self.h2_flux
            co_mass = self.co_flux
            ch4_mass = self.ch4_flux
            return 'O2:' + str(o2_mass) + ', N2:' + str(n2_mass) + ',
CH4:' + str(ch4_mass)+ ', H2:' + str(h2_mass)+ ', CO:' + str(co_mass)
```

```
    def isValidFile(self, filename):
        """
        Determines if the filename conforms to that of simulation
output.

        Input: filename
        Output: True => Valid Simulation Output
                False => Invalid File
        """
        self.retrieveNameInfo(filename)
        return self.valid_file
```

C-2: parseChemkin2

Description:

The parseChemkin2 plot script one class and a grouping of functions which allow thermodynamic data stored in Chemkin2 format to be plotted, saved in comma separated file format, and store values for the difference between the low-temperature and high-temperature polynomial results at the boundaries between temperature ranges.

The figure below shows a typical Chemkin2 format thermodynamic data entry. The actual thermodynamic data is stored in the form of two 7-coefficient NASA thermodynamic polynomials. Each of the two polynomials covers a temperature range which may be either globally specified for all entries within a file or included directly within each entry. Each of the fourteen polynomial values within an entry occupies an exact position within that entry.

```
CH4    RRHO       g 8/99C  1.H  4.   0.   0.G   200.000  6000.000 1000.        1
 1.91178600E+00 9.60267960E-03-3.38387841E-06 5.38797240E-10-3.19306807E-14    2
-1.00992136E+04 8.48241861E+00 5.14825732E+00-1.37002410E-02 4.93749414E-05    3
-4.91952339E-08 1.70097299E-11-1.02453222E+04-4.63322726E+00-8.97226656E+03    4
```

Figure C-3: Example of a Chemkin2 Format Thermodynamic Polynomial Data Entry(Burcat 2005)

Code Overview:

The parseChemkin2 code contains one class with functions and a few standalone functions. The class, ThermoData, stores species names, temperature ranges, and NASA polynomial coefficients lists for each Chemkin2 format thermodynamic data file, while the standalone functions are used to analyze and plot all species contained within the thermodynamic data file. The following tables give descriptions of constants used by the parseChemkin2 code, the ThermoData class functions, and each standalone function. These variables are intended for internal use by the parseChemkin2 code, but it is possible for users to make modification.

Constant	Value	Purpose
DONE	-1	Used to indicate that not all thermodynamic data has been read.
NOT_DONE	1	Used to indicate that all thermodynamic data has been read.
name_start	0	The character position on a row for the start of a species name.
name_end	14	The character position on a row for the end of a species name.
low_start	46	The character position on a row for the start of a low temperature range value.
low_end	55	The character position on a row for the end of a low temperature range value.
mid_start	66	The character position on a row for the start of a low temperature range value.
mid_end	75	The character position on a row for the end of a low temperature range value.
high_start	56	The character position on a row for the start of a high temperature range value.
high_end	65	The character position on a row for the end of a high temperature range value.
nCoeff	7	The number of coefficients needed for one "old" NASA polynomial for a single temperature range. Note that one thermodynamic file entry contains two sets of NASA polynomial coefficients.
R	8314.0	The ideal gas constant.
end_string	"END"	The text used to mark the end of a Chemkin2 format thermodynamic data file.
begin_string	"THERMO"	The text used to mark the beginning of a Chemkin2 format thermodynamic data file.

Table C-5: Table of Constants Used by the parseChemkin2 Code

ThermoData Class Functions				
Function	**Description**	**Inputs**	**Outputs**	**Purpose**
__init__	Used to initialize a ThermoData object			
read	Reads a single thermodynamic data entry and stores data to class variables.	infile		A pointer to the a Chemkin2 format thermodynamic data file opened for reading.
			status	Indicates if the last entry for the input Chemkin2 thermodynamic data file has been found.
getName	Gets the name of a species for which thermodynamic data has been read.	species_position		The number of the entry for which a species name is needed.
			name	The name of the species corresponding to the entry number.
getLowCoef	Retrieves the low temperature NASA thermodynamic coefficients for a specified species entry	species_position		The number of the entry for which the low temperature coefficients are needed
			coeff_low	A list containing the low temperature range NASA polynomial coefficients for the species corresponding to the input entry number.
getHighCoef	Retrieves the high temperature NASA thermodynamic coefficients for a specified species	species_position		The number of the entry for which the low temperature coefficients are needed.
			coeff_high	A list containing

	entry.			the high temperature range NASA polynomial coefficients for the species corresponding to the input entry number.
getT_low	Retrieves the low temperature value for the low temperature range of a specified Chemkin2 format thermodynamic entry.	species_position		The number of the entry for which the low value of the low temperature range is needed.
			T_low	The low temperature value for the low temperature NASA thermodynamic polynomial of the species corresponding to the input entry number
getT_mid	Retrieves the temperature value between the two temperature ranges in a specified Chemkin2 format thermodynamic entry.			The number of the entry for which the midpoint temperature range is needed.
			T_mid	The temperature value between the two temperature ranges of the NASA thermodynamic polynomials of the species corresponding to the input entry number
getT_high	Retrieves the high temperature in the high temperature range of a Chemkin2 format	species_position		The number of the entry for which the high value of the high temperature range is needed.
			T_high	The high temperature value

	thermodynmamic data entry.			for the high temperature NASA thermodynamic polynomial of the species corresponding to the input entry number
numSpecies	Gives the number of thermodynamic data entries read from the provided Chemkin2 format thermodynamic data file		num	The number of species that have been read in from the current Chemkin2 format thermodynamic data file.
calcAll	Calculates enthalpy, entropy, and specific heat at a specific temperature for a particular thermodynamic data entry.	T		Temperature in Kelvin
		species_position		The number of the entry for the species in which the properties need to be calculated.
		L_or_H		Specifies whether to use the low temperature coefficients ('L'), high temperature coefficients ('H'), or pick based upon temperature range ('U').
			values	A python tuple containing specific heat, enthalpy, and entropy in that order.
plot	Plots specific heat, entropy, and enthalpy over a specified temperature range for a specified species.	T_list		A list containing temperature values for which to compute thermodynamic properties for plots.

		species_position		The number of the thermodynamic entry to plot.
toCSV	Stores values of specific heat, enthalpy, and entropy for a specified species and set of temperatures in a comma separated value file.	T		A list of temperatures for which to calculate thermodynamic properties.
		species_position		The number of the thermodynamic entry for which calculation is needed.
midPointError	Calculates the difference in values produced from the use of the high temperature and low temperature coefficients at the point between the two ranges for a specified species.	species_position		The number of the thermodynamic entry for which the error between high and low temperature ranges is needed.
			Values	A Python tuple containing the difference between calculated value between temperature polynomials for the overlapping temperature of specific heat, enthalpy, and entropy in that order.

Table C-6: Class Methods for the ThermoData Class Contained in parseChemkin2 Code

Function	Description	Inputs	Outputs	Purpose
cleanFile	Deletes all instances of the characters '[' and ']' from a given file.	filename		The name of the file for which brackets need to be deleted.
parseThermo	Drives the reading of a Chemkin2 format thermodynamic data file until it is complete	infile		A file pointer to a Chemkin2 format thermodynamic data file.
			thermo_list	A list containing all NASA polynomial data contained in a Chemkin2 format thermodynamic data file.
parseFile	Drive the full parsing of a Chemkin2 format thermodynamic data file	filename		Name of a Chemkin2 format thermodynamic data file.
runThermo	Similar to a main function, this function calls all other functions and creates ThermoData objects to be used for plotting specific heat, enthalpy, entropy, and errors between temperature ranges.	filename		The name of the Chemkin2 format thermodynamic data file to be processed.

Table C-7: Standalone Methods Contained in the parseChemkin2 Code

Code Listing:

```
import Numeric as num
import matplotlib
matplotlib.interactive(True)
matplotlib.use('TkAgg')
from matplotlib.matlab import *

DONE = -1
NOT_DONE = 1
name_start = 0
name_end = 14
low_start = 46
low_end = 55
mid_start = 66
mid_end = 75
high_start = 56
high_end = 65
nCoeff = 7
R = 8314.0
end_string = "END"
begin_string = "THERMO"

def newVec(n, val = 0.0):
    return val*num.ones(n,'d')

class ThermoData:

    def __init__(self):

        self.coeff_low = []
        self.T_low = []
        self.T_mid = []
        self.T_high = []
        self.coeff_high = []
        self.name = []

    def read(self, infile):
        """reads the approriate lines from chemkin thermodynamic data
file
        and fills the high and low coefficient arrays.

        if the end of the section has been reached return -1
        """

        # read header
        text = infile.readline()
        if (end_string in text) or (end_string.swapcase() in text):
            return DONE
        elif (begin_string in text) or (begin_string.swapcase() in
text):
            text = infile.readline()
            try:
                temp = float(text[0:10])
            except:
                # if not a float, then there is no global range line
```

```
                # do nothing
                two = 2
            else:
                text = infile.readline()
                # if temp is a float, then there is a global range line
                # Skip past that line


        # extract species name and temperatures
        self.name.append(text[name_start:name_end+1])
        self.T_low.append(float(text[low_start:low_end+1]))
        self.T_mid.append(float(text[mid_start:mid_end+1]))
        self.T_high.append(float(text[high_start:high_end+1]))

        # read coefficient lines
        nLines = 3                  # number of lines of coefficient in
the data file
        nCoeffsPerLine = 5          # number of coefficient per line
        nCoeffStringLength = 15     # length of the coefficient string
        coeffs = []
        for k in range(nLines):
            line = infile.readline()
            for j in range(nCoeffsPerLine):
                start = j*nCoeffStringLength
                start_next = start + nCoeffStringLength
                try:
                    coeffs.append([float( line[start:start_next] )])
                except:
                    two = 2

        # extract high and low temperater coefficients
        self.coeff_low.append(list(coeffs[0:nCoeff]))
        self.coeff_high.append(list(coeffs[nCoeff:2*nCoeff]))
        return NOT_DONE

    def getName(self, species_position):
        return self.name[species_position]
    def getLowCoef(self, species_position):
        return self.coeff_low[species_position]
    def getHighCoef(self, species_position):
        return self.high_coeffs[species_position]
    def getT_low(self, species_position):
        return self.T_low[species_position]
    def getT_mid(self, species_position):
        return self.T_mid[species_position]
    def getT_high(self, species_position):
        return self.T_high[species_position]
    def numSpecies (self):
        """Returns the number of species read"""
        return len(self.name)


    def calcAll(self, T, species_position, L_or_H = 'U'):
        """calculate specific heat, enthalpy and entropy.
        L_or_H may be set to 'L' for low range coefs,
                             'H' for high range coefs, or
```

```
                            'U' in order to pick based on the
temperature value"""
        M1 = [1.0, float(T), float(T**2), float(T**3), float(T**4),
0.0, 0.0]
        M2 = [1.0, float(T)/2.0, float(T**2)/3.0, float(T**3)/4.0,
float(T**4)/5.0, 1.0/float(T), 0.0]
        M3 = [log(float(T)), float(T), float(T**2)/2.0,
float(T**3)/3.0, float(T**4)/4.0, 0.0, 1.0]
        if L_or_H == 'U' or L_or_H == 'u':
            if T < self.T_mid:
                a = self.coeff_low[species_position]
            else:
                a = self.coeff_high[species_position]
        elif L_or_H == 'L' or L_or_H == 'l':
            a = self.coeff_low[species_position]
        elif L_or_H == 'H' or L_or_H == 'h':
            a = self.coeff_high[species_position]

        cp = num.matrixmultiply(M1, a)*R
        H = num.matrixmultiply(M2, a)*(R*float(T))
        S = num.matrixmultiply(M3, a)*R
        return cp,H,S

    def toCSV(self, T, species_position):
        """write cp,h and s to a csv file."""
        n = len(T)
        csv = open(self.name[species_position].split(' ')[0] + '.csv',
'w')
        csv.write('species: ' + self.name[species_position] + '\n')
        csv.write('Temperature,cp,H,S\n')
        for i in range(n):
            if T[i] == self.T_mid[species_position]:
                cp,h,s = self.calcAll(T[i], species_position, 'L')
                csv.write(str(T[i]) + ',' + str(cp) + ',' +str(h) + ','
+ str(s) + '\n')
                cp,h,s = self.calcAll(T[i], species_position, 'H')
                csv.write(str(T[i]) + ',' + str(cp) + ',' +str(h) + ','
+ str(s) + '\n')
            else:
                cp,h,s = self.calcAll(T[i], species_position)
                csv.write(str(T[i]) + ',' + str(cp) + ',' +str(h) + ','
+ str(s) + '\n')
        csv.close()
        cleanFile(self.name[species_position].split(' ')[0] + '.csv')

    def plot(self, T_list, species_position):
        """plot cp,h and s"""
        n = len(T_list)
        cp = []
        S = []
        H = []
        for i in range(n):
            temp1, temp2, temp3 = self.calcAll(T_list[i],
species_position)
            cp.append(float(list(temp1)[0]))
            S.append(float(list(temp2)[0]))
            H.append(float(list(temp3)[0]))
```

```
        clf()
        plot(T_list, cp)
        title(self.name[species_position].split(' ')[0] + ' cp')
        xlabel('Temperature')
        ylabel('Specific Heat')
        savefig('./' + self.name[species_position].split(' ')[0] +
'_cp.png')

        clf()
        plot(T_list, S)
        title(self.name[species_position].split(' ')[0] + ' H')
        xlabel('Temperature')
        ylabel('Enthalpy')
        savefig('./' + self.name[species_position].split(' ')[0] +
'_H.png')

        clf()
        plot(T_list, H)
        title(self.name[species_position].split(' ')[0] + ' S')
        xlabel('Temperature')
        ylabel('Entropy')
        savefig('./' + self.name[species_position].split(' ')[0] +
'_S.png')

    def midPointError(self, species_position):
        """calculates the difference in polynomials at the mid-point"""
        h_cp, h_H, h_S = self.calcAll(self.T_mid[species_position],
species_position, 'H')
        l_cp, l_H, l_S = self.calcAll(self.T_mid[species_position],
species_position, 'L')
        err_cp = h_cp - l_cp
        err_H  = h_H - l_H
        err_S  = h_S - l_S
        return err_cp, err_H, err_S

def cleanFile(filename):
    f = open(filename, 'r')
    temp = open('temp', 'w')

    while(True):
        text = f.read(1)
        if text == '':
            break
        if text != '[' and text != ']':
            temp.write(text)
    f.close()
    temp.close()
    f=open(filename, 'w')
    temp=open('temp','r')

    while(True):
        text = temp.read(1)
        if text == '':
            break
        f.write(text)
    f.close()
```

```
    temp.close()


def parseThermo(infile):
    """reads thermodynamic section for all the species in a Chemkin
format
    data file and returns a list of ThermoData objects"""

    thermo_list = []
    while(True):
        data = ThermoData(infile)
        if ( data.read() == NOT_DONE ):
                thermo_list.append( data )
        else:
            break

    return thermo_list

def parseFile(filename):
    """parse a Chemkin format thermodynamic file"""

    # file is read/parsed line by line
    # an alternative would be to read the entire file using read then
split lines using
    # the string method split

    comment = "!"
    start_string = "THERMO"
    end_string = "END"

    f = open(filename,'r')
    logfile = open(filename,'w')
    # search file for keyword

    line = "stuff"
    thermo_list
    while(len(line) > 0):
        line = f.readline()
        if (start_string in line) or (start_string.swap_case() in
line):
            thermo_list = parseThermo(f)

    # use thermo_data list to
    #    generate csv files
    #    generate plots
    #    check mid-point errors

    T = num.arange(300.0,3100.0,100.0)
    for thermo_data in thermo_list:
        thermo_data.toCSV(T)
        thermo_data.plot(T)
        print thermo_data.name,thermo_data.midPointError()

def runThermo(filename):
    """Generates files containing plots and tables of cp, H, and S for
each species,
```

```
    and the midpoint error for all species contained in a old NASA
format thermodynamic
    data file."""
    therm = ThermoData()

    input = open(filename, 'r')

    while(therm.read(input) == NOT_DONE):
        #read each species from thermo file
        two = 2

    input.close()
    mpr = open('mp_err.csv', 'w') #file for midpoint errors
    mpr.write("Species, Err_cp, Err_H, Err_S, Midpoint\n")

    T = range(300.0, 5025.0, 25.0)

    for i in range(0,therm.numSpecies()):
        therm.toCSV(T,i)
        therm.plot(T,i)
        err_cp, err_H, err_S = therm.midPointError(i)
        mpr.write(therm.getName(i) + ',' + str(err_cp) + ',' +
str(err_H) + ',' + str(err_S) + ',' + str(therm.getT_mid(i)) + '\n')

    mpr.close()

    cleanFile('mp_err.csv')
```

Appendix D: Output File Formats

D-1: CSV Simulation Output File

The Comma Separated Value Simulation Output File is provided for its compatibility with existing spreadsheet software. As the name implies, this file format stores a series of values separated by commas within each row. Additional rows are produced by simply adding additional lines in the file. The contents are generated from a solved flame object and stored as columns in a 'csv' file. These contents include grid points, gas velocity, rate, temperature, voltage (if used), and mole fractions of all species at each grid point. Data labels are provided in the first row of the file. Figure E-1 gives a truncated example this file type.

```
Z (m), U(m/s), V (1/s), T (K), Voltage, CH4,...
0.0, 0.1218099, 0.0, 500.0, 2.784113e-19, 0.02362278,...
6.25000e-06, 0.125886, 0.0, 516.742642, 0.03868586, 0.02275284,...
1.25000e-05, 0.129823, 0.0, 532.911221, 0.07639685, 0.02189493,...
1.87500e-05, 0.133635, 0.0, 548.569255, 0.11264910, 0.02104841,...
...
```

Figure D-1: CSV Simulation Output File Truncated Example

Although this file may be difficult to read in a standard word processor, most common spreadsheet software, including Excel, will place each entry into cells for simplified viewing, plotting, and analysis.

Within the simulation driver programs which generated this output file type, the name is generated using the nameParse class, which is fully documented in Appendix C.

D-2: CTML Simulation Output File

A CTML Simulation Output File is automatically generated from Cantera's BurnerFlame and BurnerFlame_efield object using the objects 'save' function. This file format uses XML style flags and entries to store listings of much of the essential properties of the flame object including grid points, velocity, temperature, and species mass fractions.

Flame property values may be retrieved within a program by creating a BurnerFlame or BurnerFlame_efield object and using the objects 'restore' function to load values from the 'ctml' file into the newly created Flame object. Despite the ability to restore data from a 'ctml' file, not enough data is restored to directly rerun a simulation from the 'ctml' file contents.

Figure D-2 gives a truncated example of a 'ctml' format file. The file contains a series of tags which mark various data structures within the file. These data structure make use of a starting tag which includes the tag type and any additional arguments between a set of angle brackets and an ending tag which includes the tag type with a leading '/' between angle brackets. The ctml tags located at the beginning and end of the file mark the entire file as a ctml document. The next major tag is the simulation tag, which marks all the data included in one simulation. There may be multiple simulation tags within a file, each pair representing a single simulation, but each simulation will contain a unique 'id' attribute. Next, each simulation contains two string tags, one for the timestamp which saves the data and time of the simulation and a second tag for an optional description. Also within the simulation object is a set of three domains, which include the inlet, STFlow or STFlowEfield, and outlet. Each of these domain types is

specified by the type attribute. The inlet domain stores the mass flux and temperature of the burner. The STFlow or STFlowEfield domains contain the grid, flame attributes, species concentrations in mass fractions stored in a series of floatarray objects inside a grid_data object, and the gas pressure stored in a pressure tag. The outlet domain contain simply marks the flame outlet and contains no other useful data.

```
<ctml>
  <simulation id="solution">
    <string title="timestamp">
    Tue Jul  5 12:12:18 2005
    </string>
    <string title="description">none</string>
    <domain components="2" id="burner" points="1" type="inlet">
      <mdot max="100000" min="-100000">8.665627244E-02</mdot>
      <temperature max="100000" min="200">5.000000000E+02</temperature>
    </domain>
    <domain components="25" id="flame" points="187" type="StFlowEfield">
      <grid_data>
        <floatArray size="187" title="z" type="length" units="m">
          0.000000000E+00,   1.000000000E-06,   2.000000000E-06,
          ...
          1.600000000E-02
        </floatArray>
        <floatArray size="187" title="u" type="velocity" units="m/s">
          1.285366700E-01,   1.291978076E-01,   1.298570238E-01,
          ...
          3.178068194E-01,   3.178068196E-01,   3.178068196E-01,
          3.178068196E-01
        </floatArray>
        <floatArray size="187" title="V" type="rate" units="1/s">
          0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
          ...
          0.000000000E+00
        </floatArray>
        <floatArray min="0" size="187" title="T" type="temperature" units="K">
          5.000000000E+02,   5.025711171E+02,   5.051347938E+02,
          ...
          1.253727606E+03,   1.253727606E+03,   1.253727606E+03,
          1.253727606E+03
        </floatArray>
        <floatArray size="187" title="L" units="N/m^4">
          0.000000000E+00,   0.000000000E+00,   0.000000000E+00,
          ...
          0.000000000E+00
        </floatArray>
          <floatArray max="1" min="0" size="187" title="CH4"
type="massFraction">
            1.390143624E-02,   1.383396936E-02,   1.376662268E-02,
          ...
          3.940754355E-28
        </floatArray>
        ...
        <floatArray size="187" title="C[e]" type="unit-charge concentration" units="kmol/m^3">
          0.000000000E+00,   3.179747343E-13,   6.077738865E-13,
          ...
          0.000000000E+00
        </floatArray>
      </grid_data>
      <pressure type="pressure" units="Pa">1.013250000E+05</pressure>
    </domain>
    <domain components="1" id="outlet" points="1" type="outlet"/>
  </simulation>
</ctml>
```

Figure D-2: CTML Output File Truncated Example

Although it is not mandatory, the filenames for this type of output file have been created using the nameParse class which is documented in Appendix C.

D-3: Running Current Output File

The Running Current Output File is a file which may be optionally produced by the EFSim_NG_Current or EFSim_Syngas_Current program which are documented in Appendix A. These files store data concerning simulation flow rates, applied voltage, and current produced by the 'current' function provided by a BurnerFlame_efield object. The name is specified by the 'current_file' input to the constructors of the EFSim_NG_Current, EFSim_NG_Current_mass, and EFSim_Syngas_Current classes.

The Running Currrent Output File stores its data in Comma Seperated Value (CSV) format which separates values within each row with a comma, and allows for multiple rows by creating additional lines. CSV are easily read by most spreadsheet software. This particular stores equivalence ratio, total flux, air flux, fuel flux, applied voltage and current for methane and air simulation. For synthesis gas and air simulations equivalence ratio, total flux, air flux, carbon monoxide flux, hydrogen gas flux, methane flux, fuel nitrogen gas flux, applied voltage and current are stored. Figures D-3 and D-4 give examples of these output file formats.

```
ER,total_flux,air_flux,fuel_flux,voltage,current
0.94266303,0.000227418429,0.000215483333,1.193509604e-05,0.0,-0.000356916694486
0.94266303,0.000227418429373,0.000215483333333,1.193509604e-05,0.1,-0.000423609085041
0.94266303,0.000227418429373,0.000215483333333,1.193509604e-05,0.2,-0.000508797391377
...
```

Figure D-3: Example of Running Current Output File Using Methane and Air

```
ER,total_flux,air_flux,CO_flux,H2_flux,CH4_flux,N2_flux,voltage,current
0.5633881,0.00023008569,0.000197160766,2.358678636e-05,1.3724605e-06,4.37154e-07,7.528525e-06,0.0,-1.53849575e-05
0.5633881,0.00023008569,0.000197160766,2.358678636e-05,1.3724605e-06,4.37154e-07,7.528525e-06,0.1,-1.85000574e-05
0.5633881,0.00023008569,0.000197160766,2.358678636e-05,1.3724605e-06,4.37154e-07,7.528525e-06,0.2,-2.15547048e-05
0.5633881,0.00023008569,0.000197160766,2.358678636e-05,1.3724605e-06,4.37154e-07,7.528525e-06,0.3,-2.40812775e-05
…
```

Figure D-4: Example of Running Current Output File Using Synthesis Gas and Air

Appendix E: Plots of Results

E-1: V-I Plots of Methane Combustion Using GRI Mech 3.0

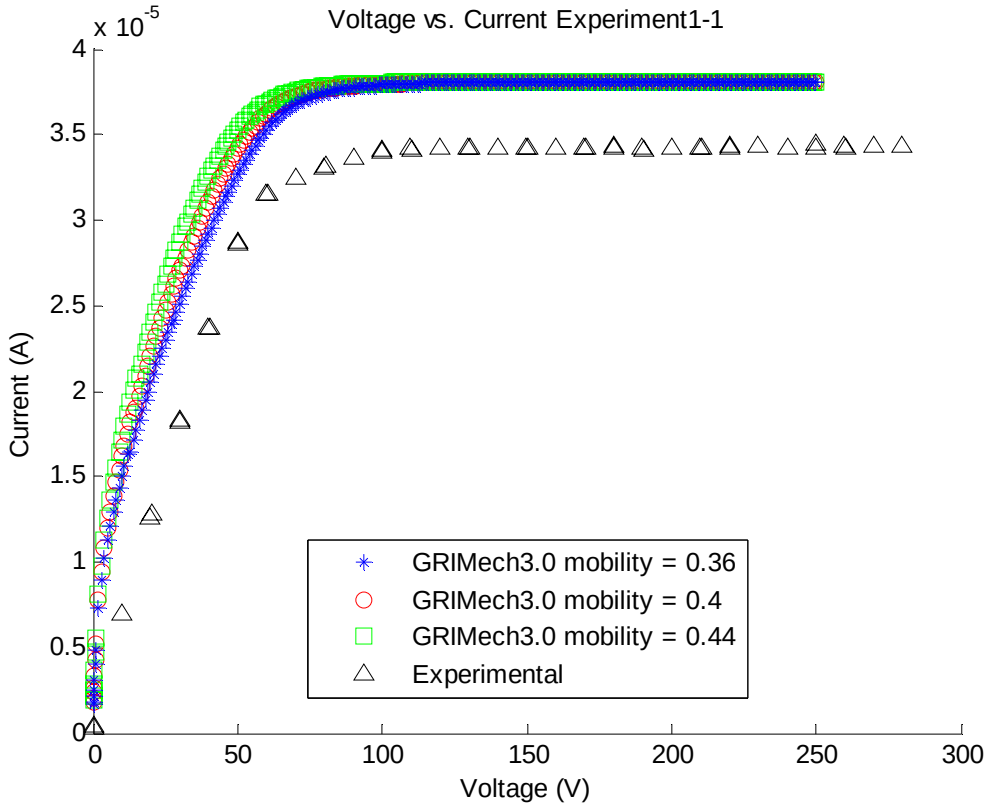

Figure E-1: Voltage vs. Current with Methane for GRI Experiment 1-1

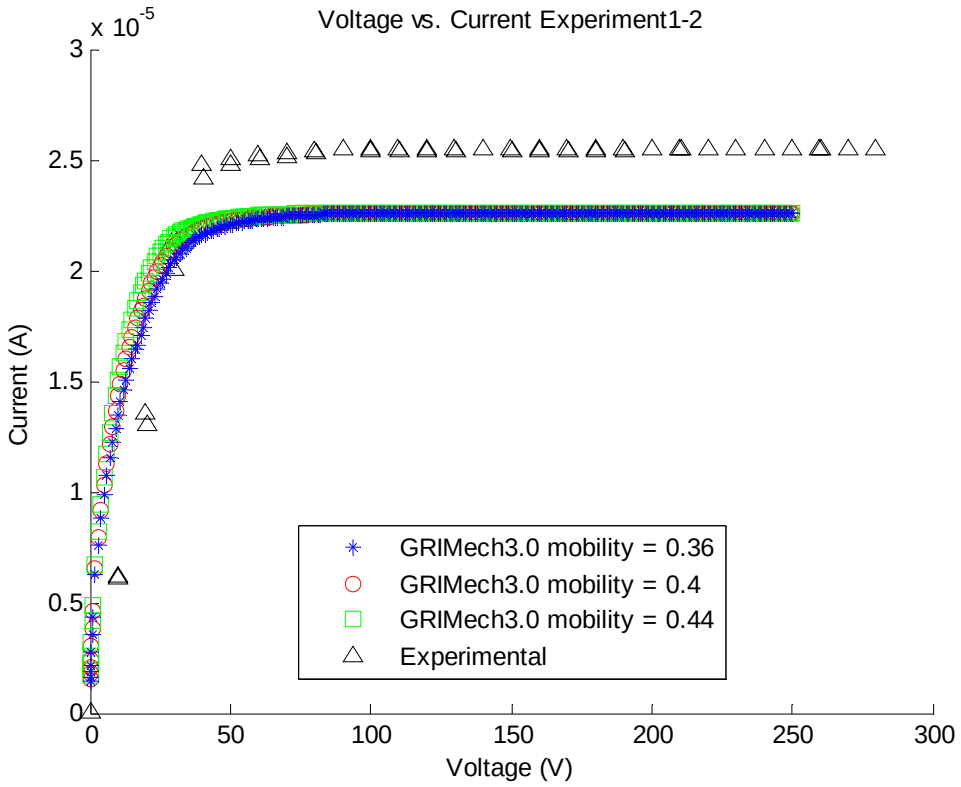

Figure E-2: Voltage vs. Current with Methane for GRI Experiment 1-2

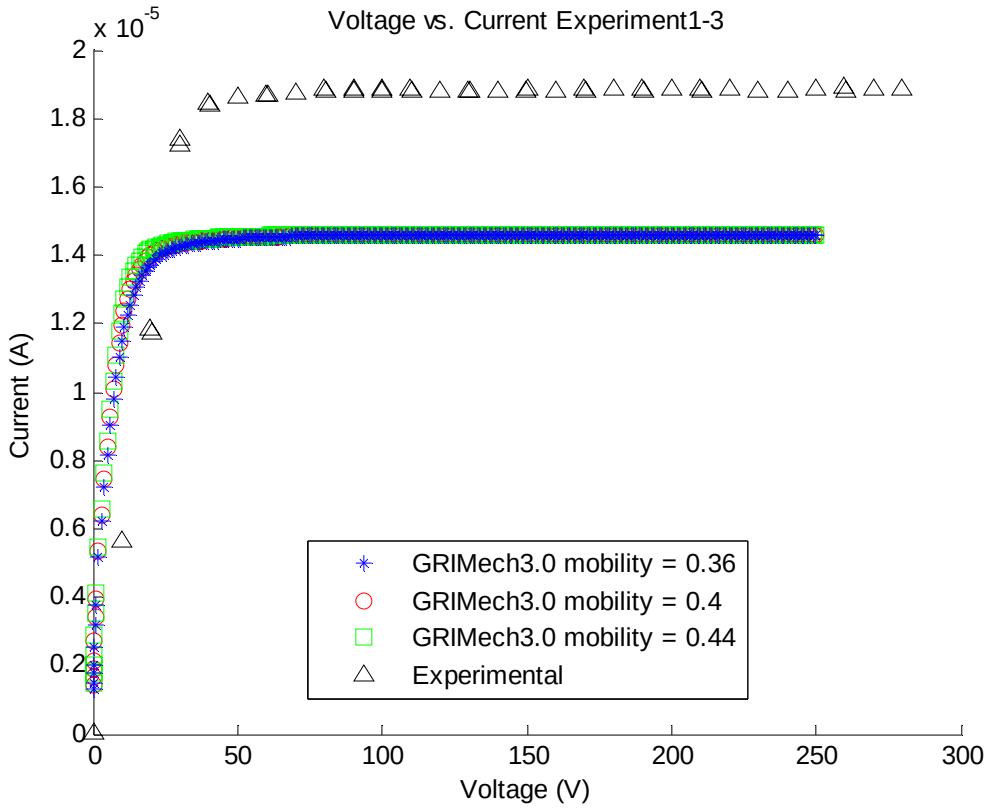

Figure E-3: Voltage vs. Current with Methane for GRI Experiment 1-3

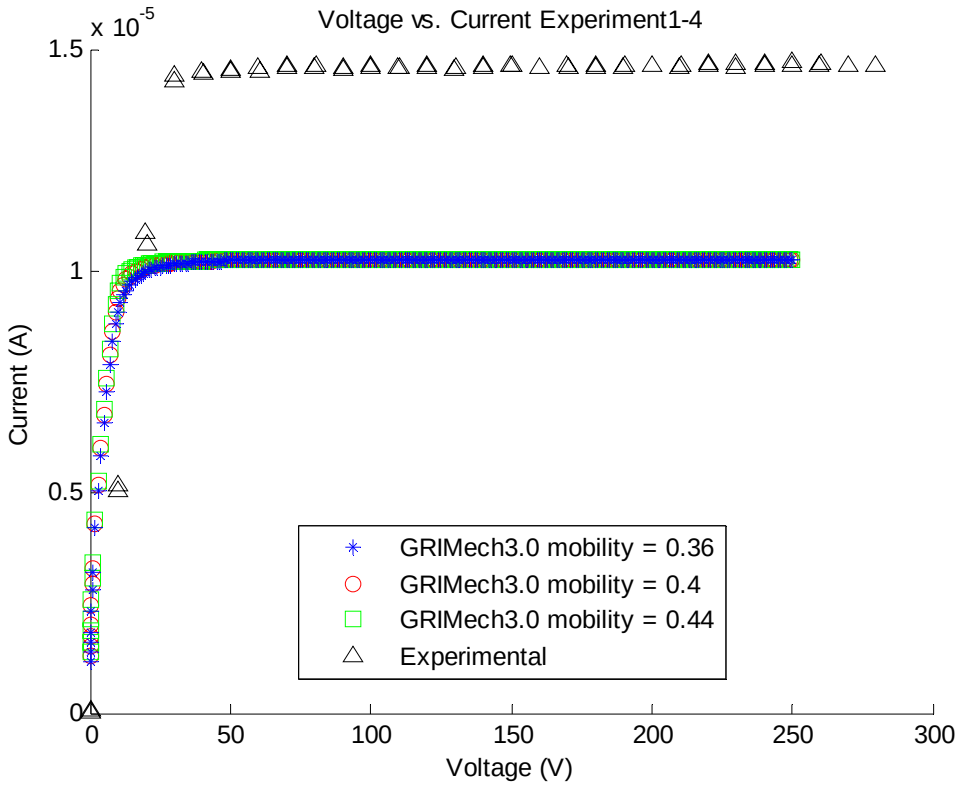

Figure E-4: Voltage vs. Current with Methane for GRI Experiment 1-4

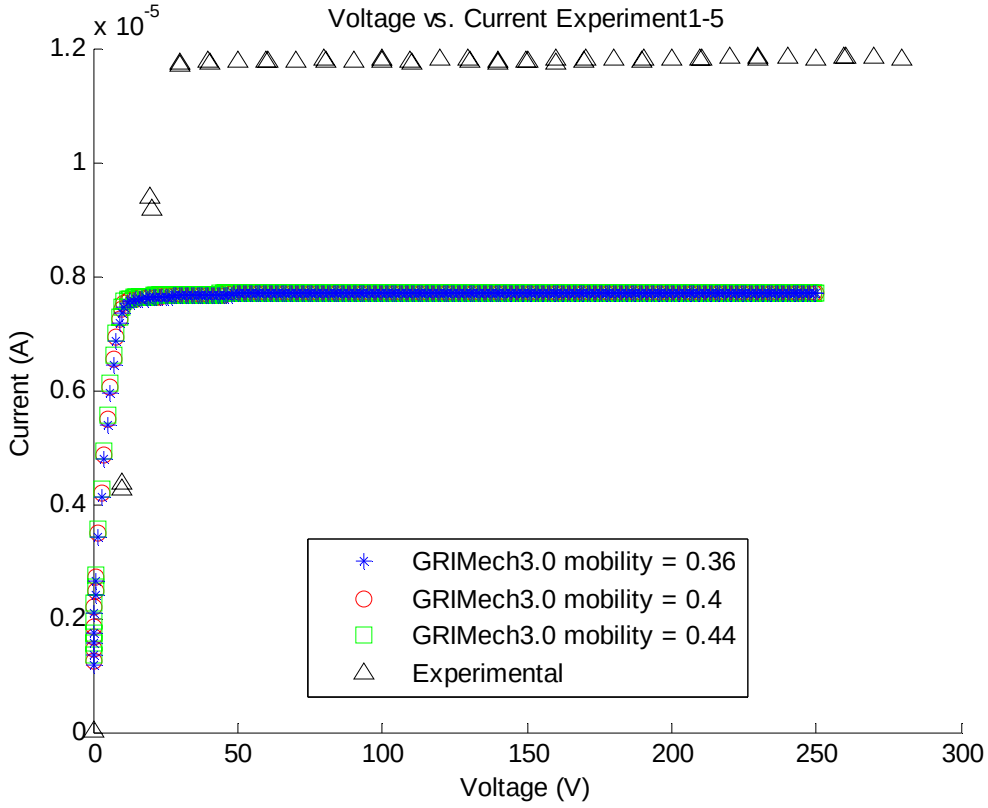

Figure E-5: Voltage vs. Current with Methane for GRI Experiment 1-5

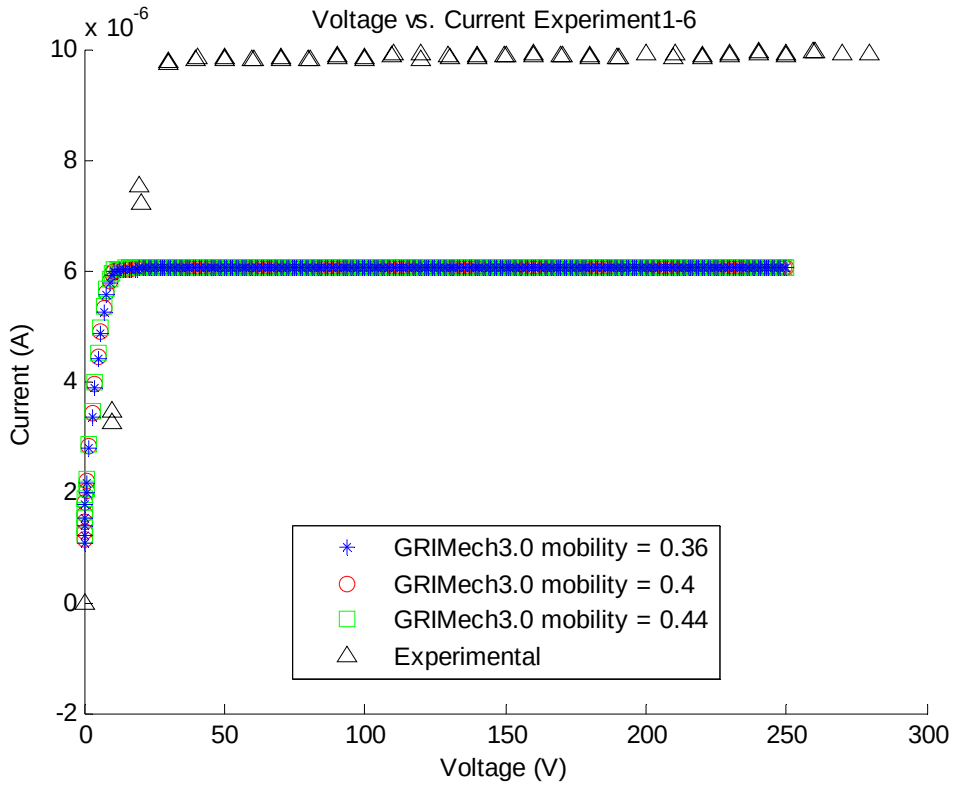

Figure E-6: Voltage vs. Current with Methane for GRI Experiment 1-6

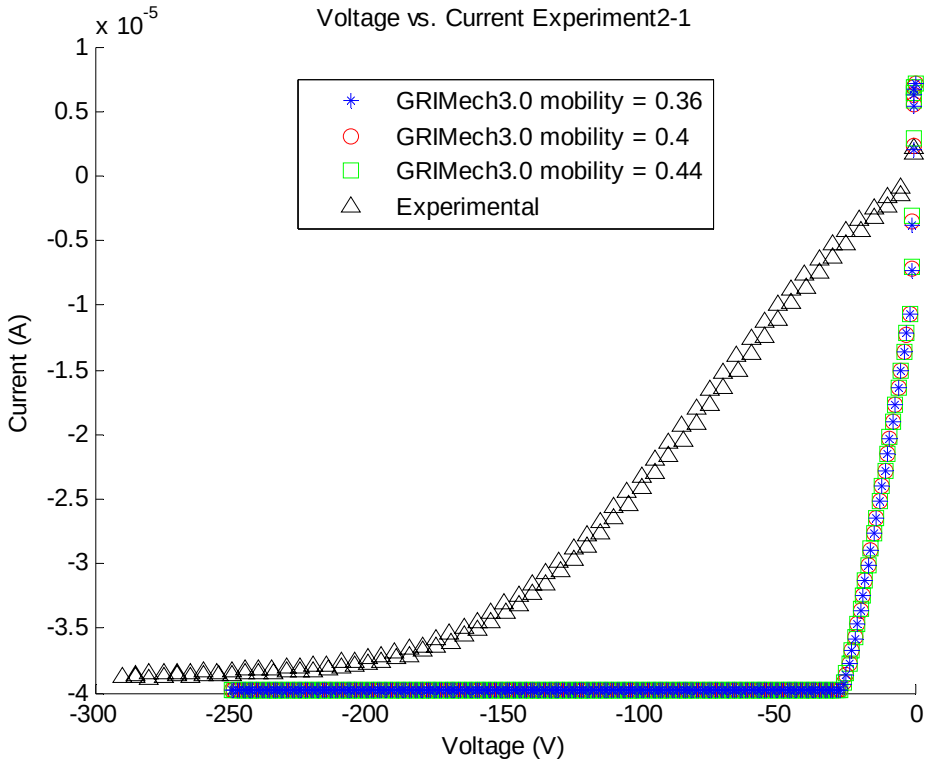

Figure E-7: Voltage vs. Current with Methane for GRI Experiment 2-1

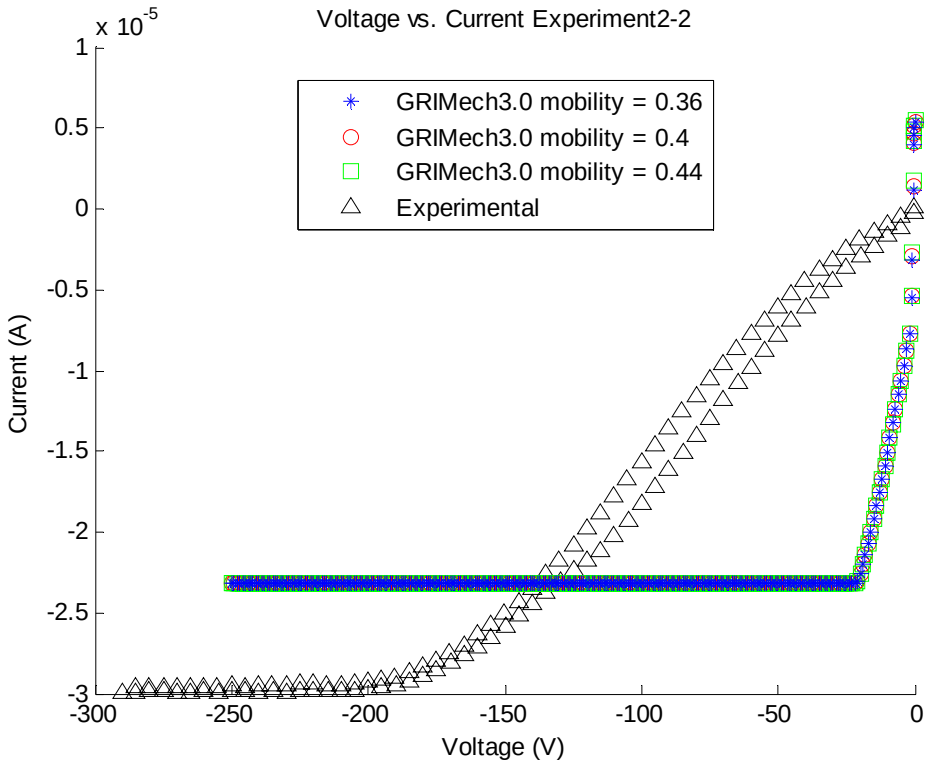

Figure E-8: Voltage vs. Current with Methane for GRI Experiment 2-2

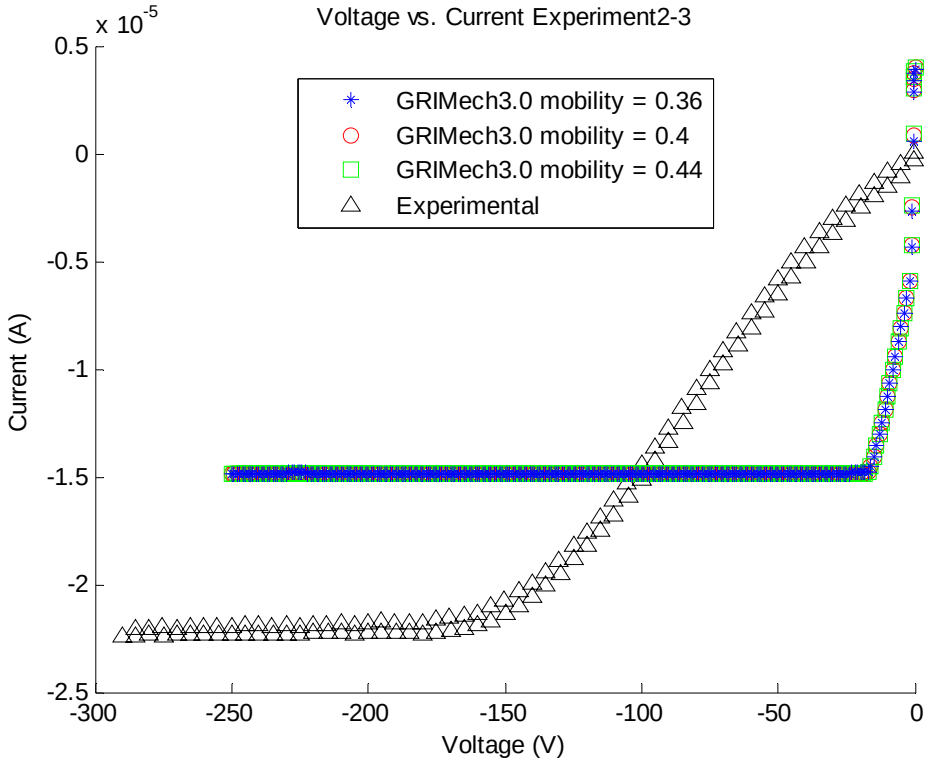

Figure E-9: Voltage vs. Current with Methane for GRI Experiment 2-3

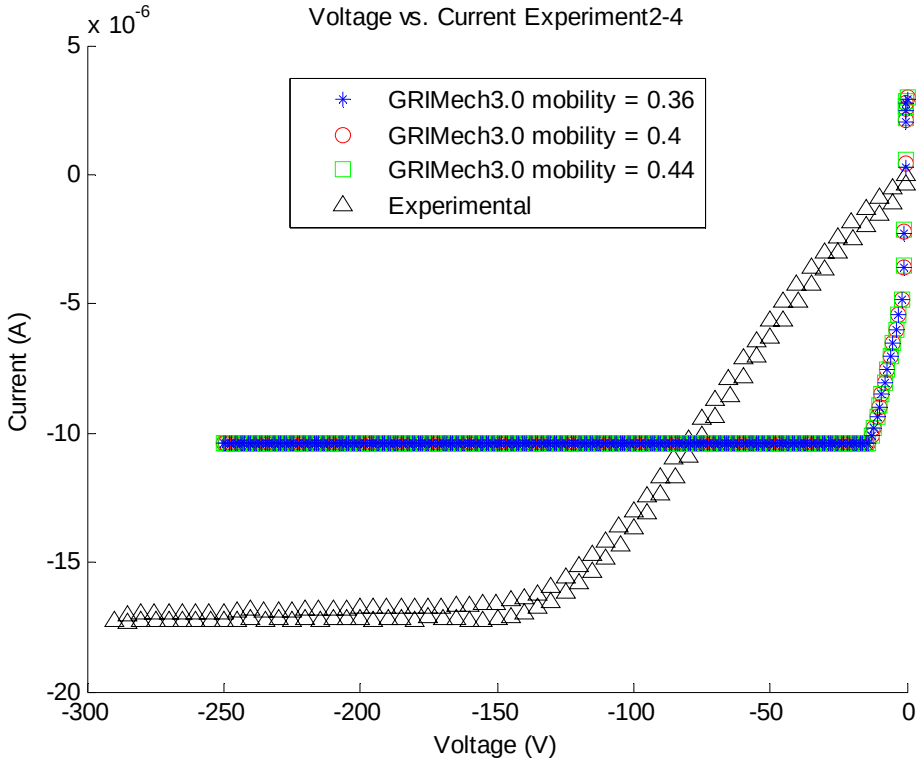

Figure E-10: Voltage vs. Current with Methane for GRI Experiment 2-4

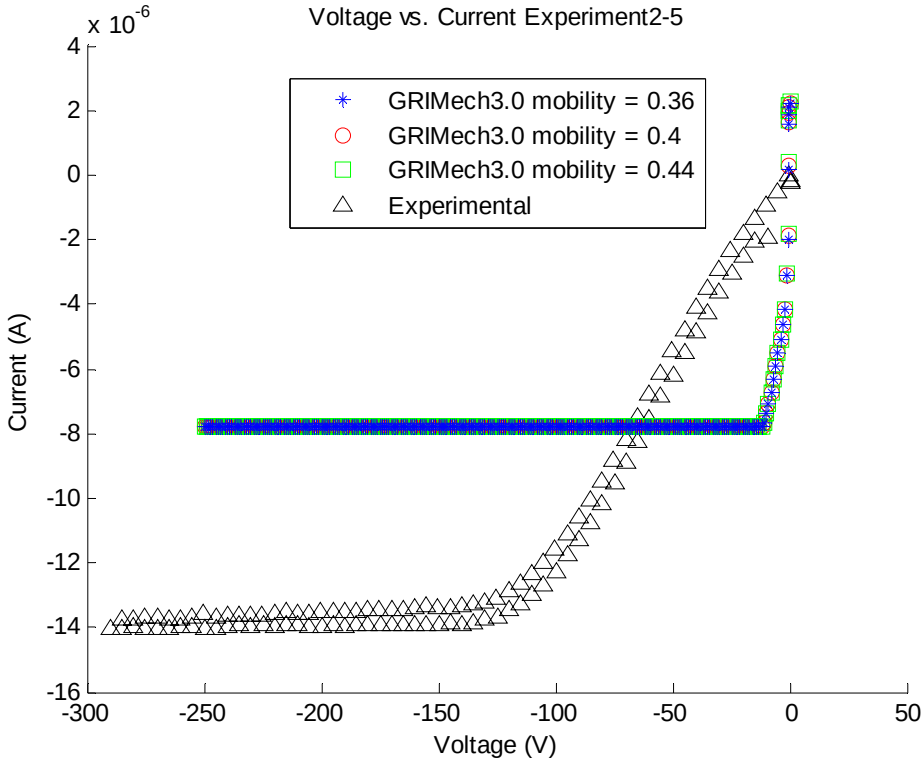

Figure E-11: Voltage vs. Current with Methane for GRI Experiment 2-5

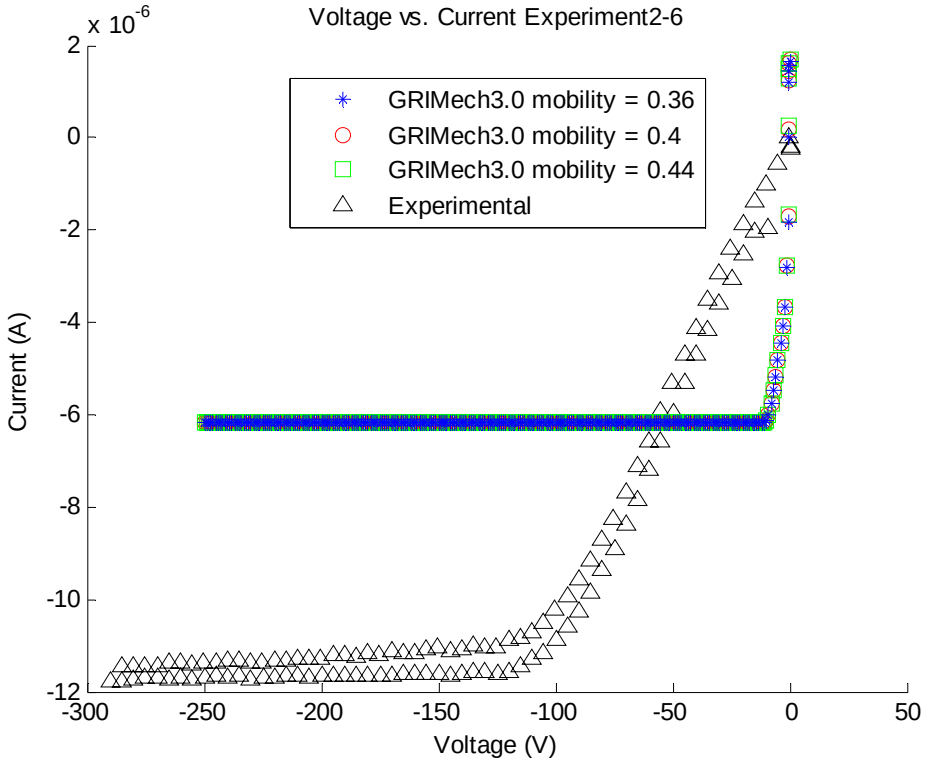

Figure E-12: Voltage vs. Current with Methane for GRI Experiment 2-6

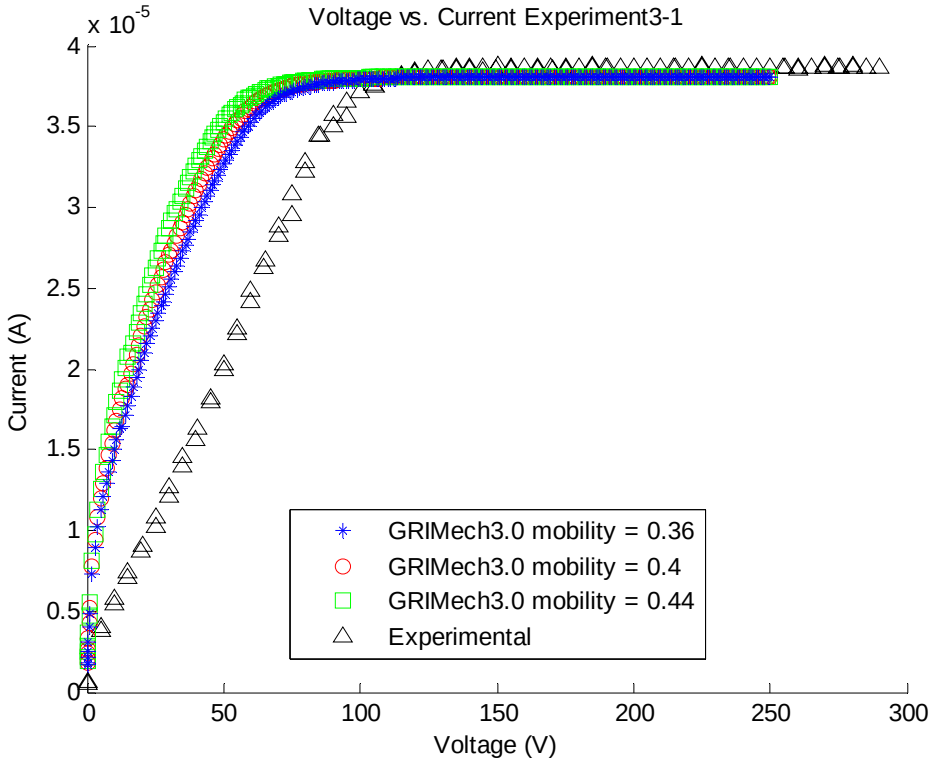

Figure E-13: Voltage vs. Current with Methane for GRI Experiment 3-1

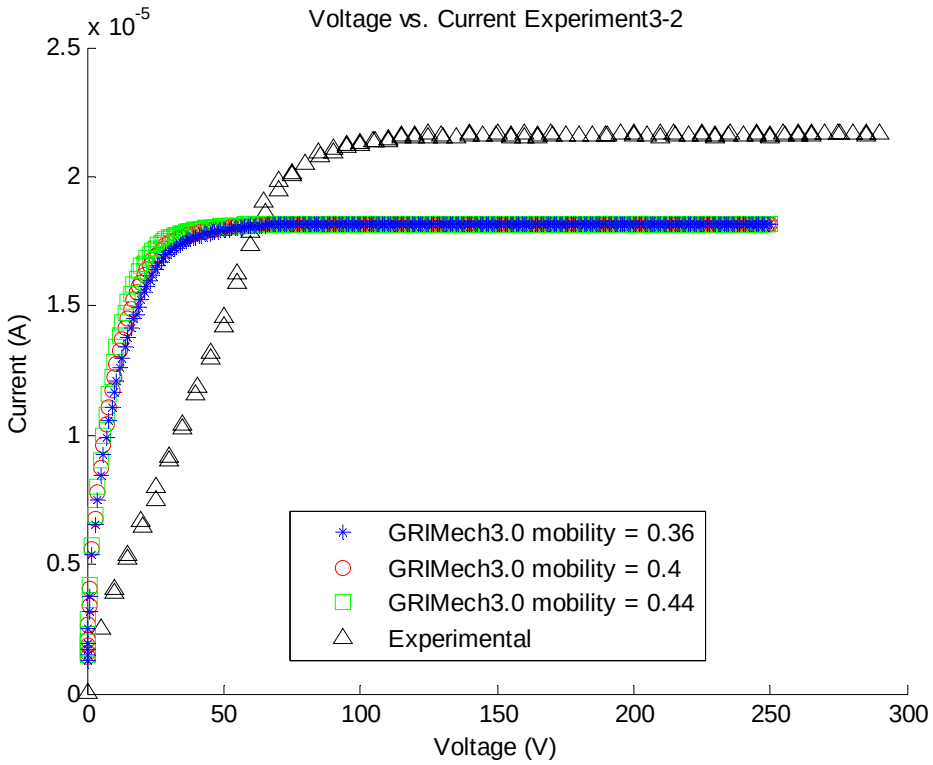

Figure E-14: Voltage vs. Current with Methane for GRI Experiment 3-2

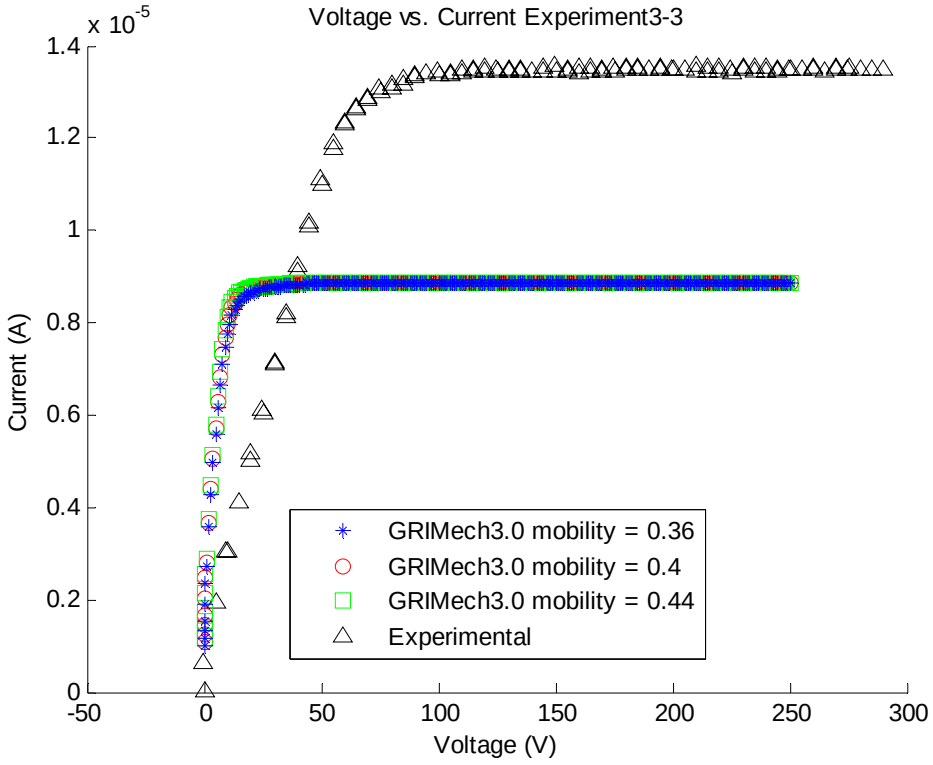

Figure E-15: Voltage vs. Current with Methane for GRI Experiment 3-3

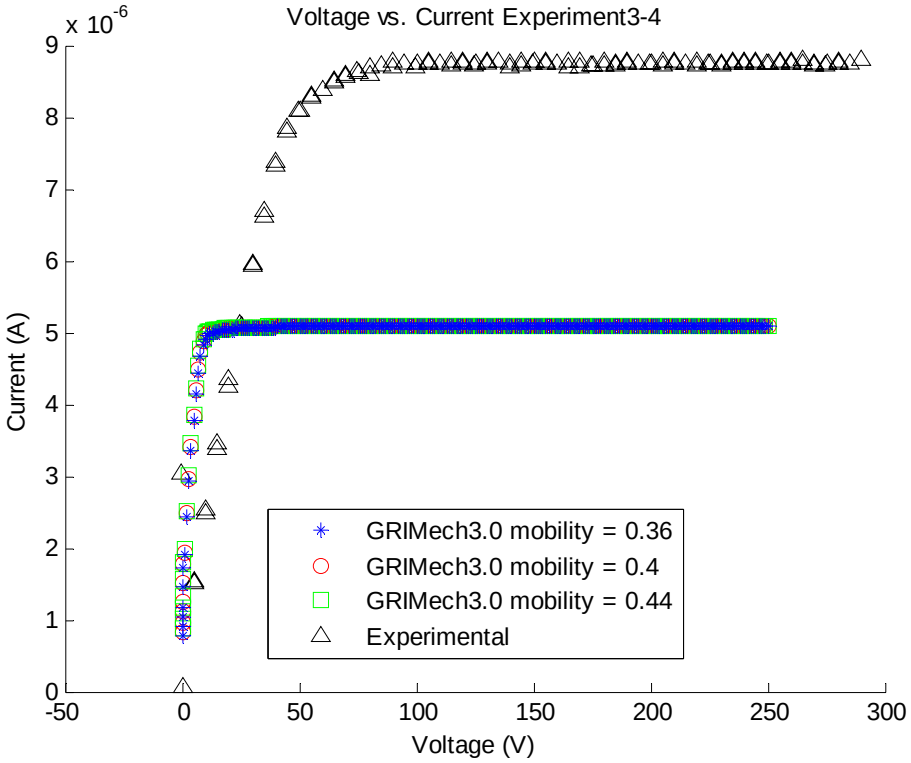

Figure E-16: Voltage vs. Current with Methane for GRI Experiment 3-4

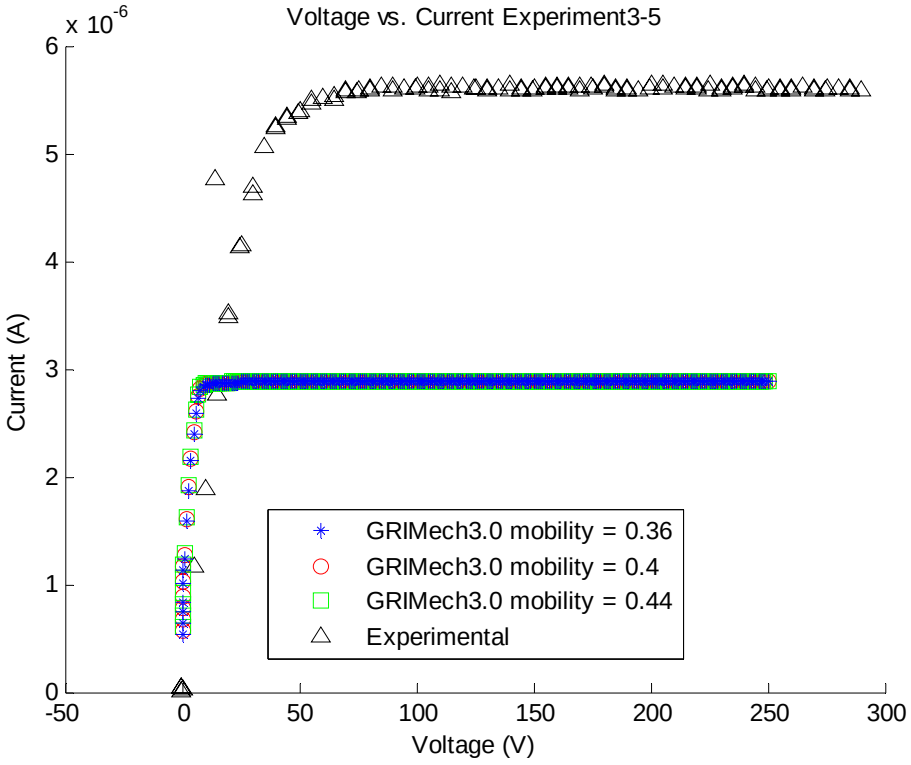

Figure E-17: Voltage vs. Current with Methane for GRI Experiment 3-5

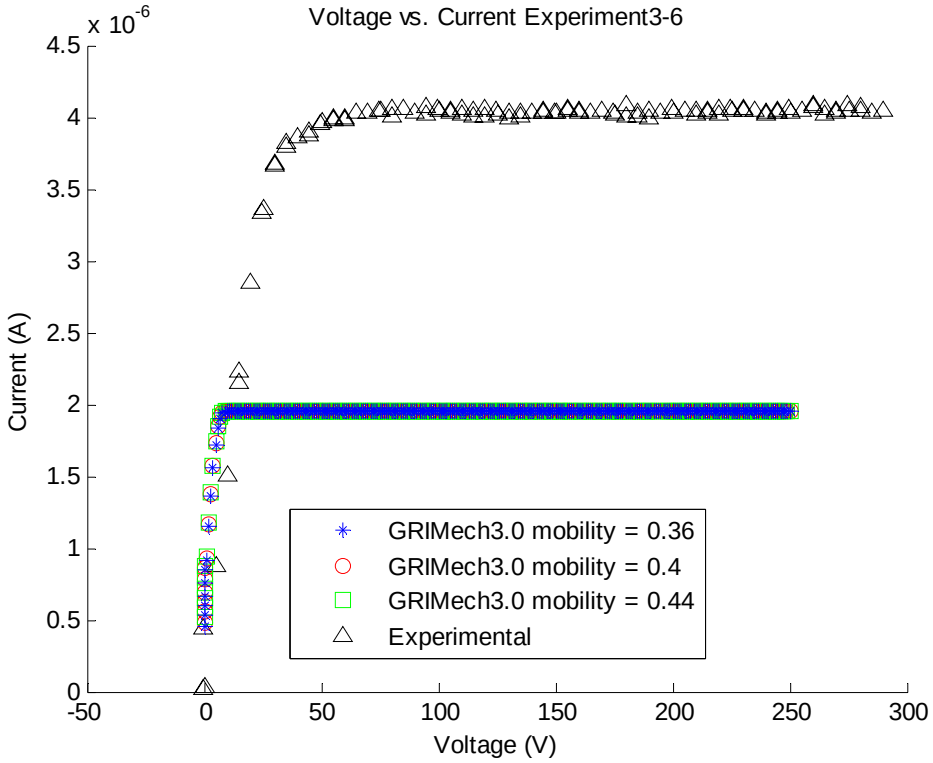

Figure E-18: Voltage vs. Current with Methane for GRI Experiment 3-6

E-2: V-I Plots of Methane Combustion Using Jones, Becker and Heinsohn

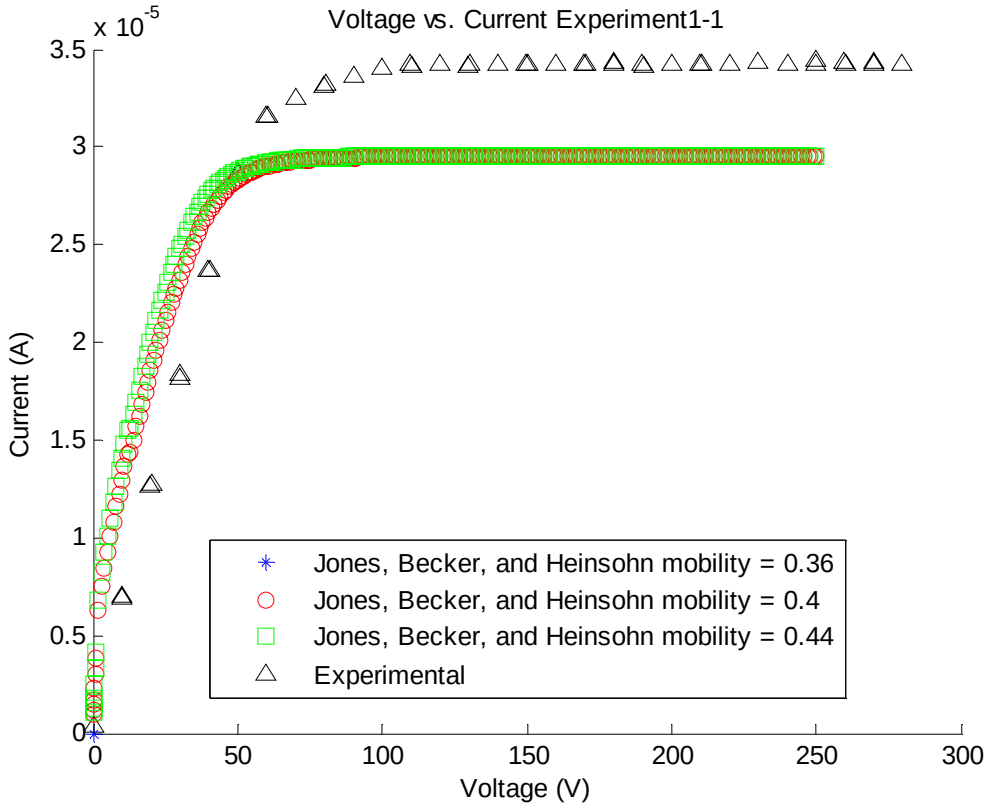

Figure E-19: Voltage vs. Current with Methane for Jones, Becker and Heinsohn Experiment 1-1

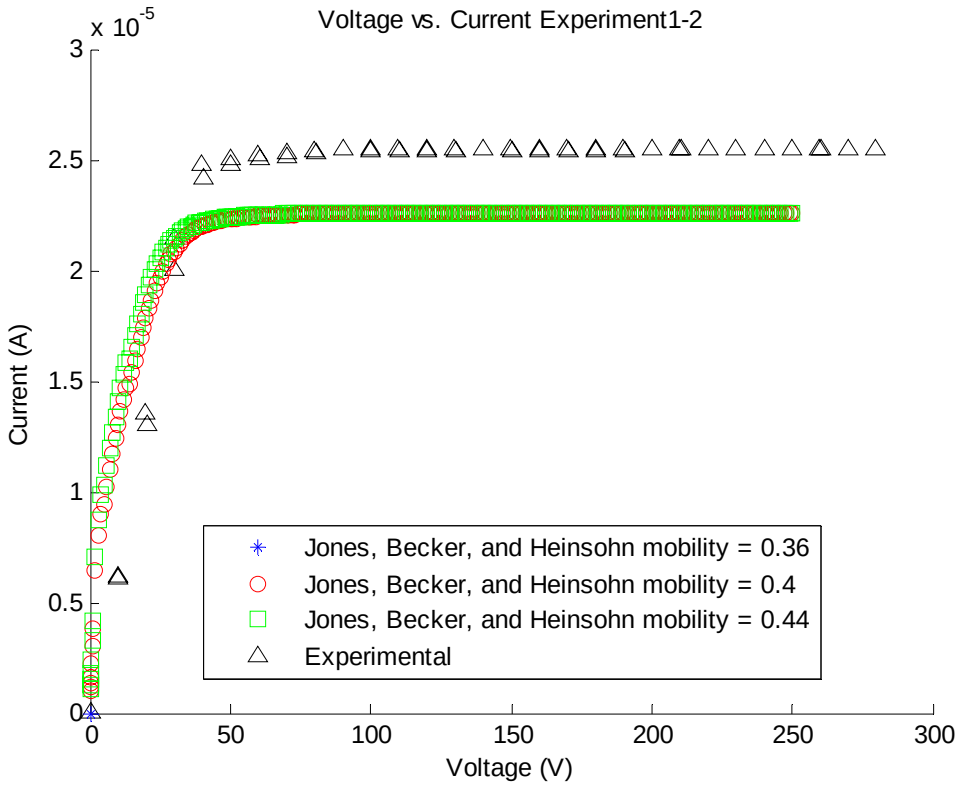

Figure E-20: Voltage vs. Current with Methane for Jones, Becker and Heinsohn Experiment 1-2

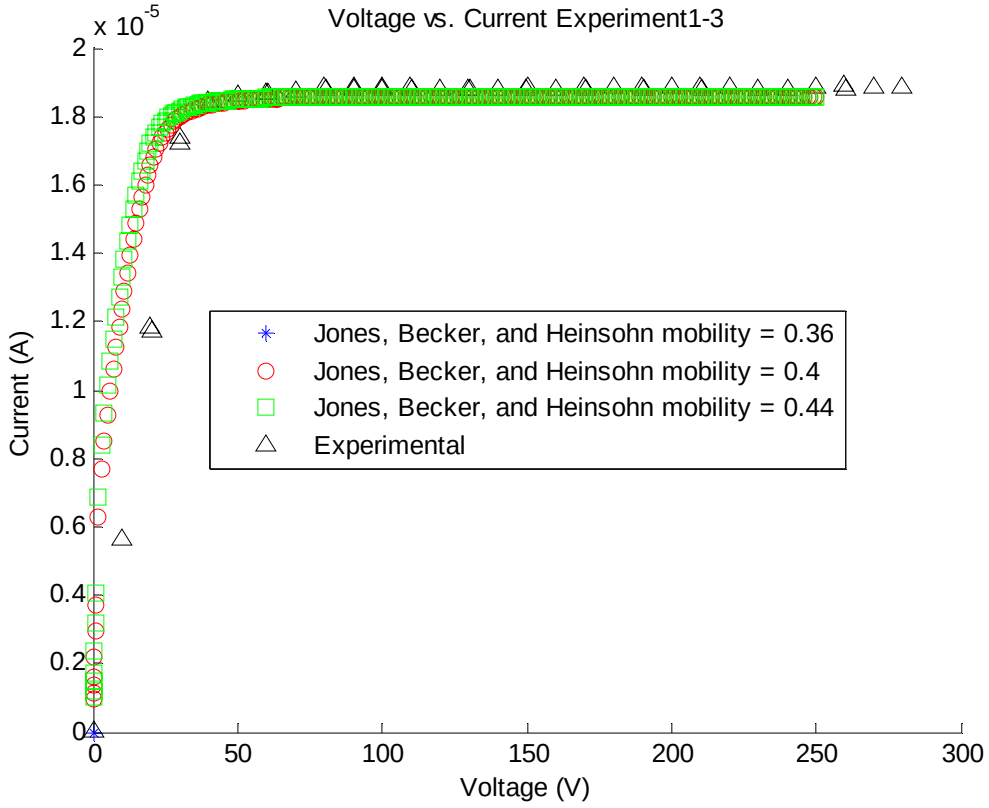

Figure E-21: Voltage vs. Current with Methane for Jones, Becker and Heinsohn Experiment 1-3

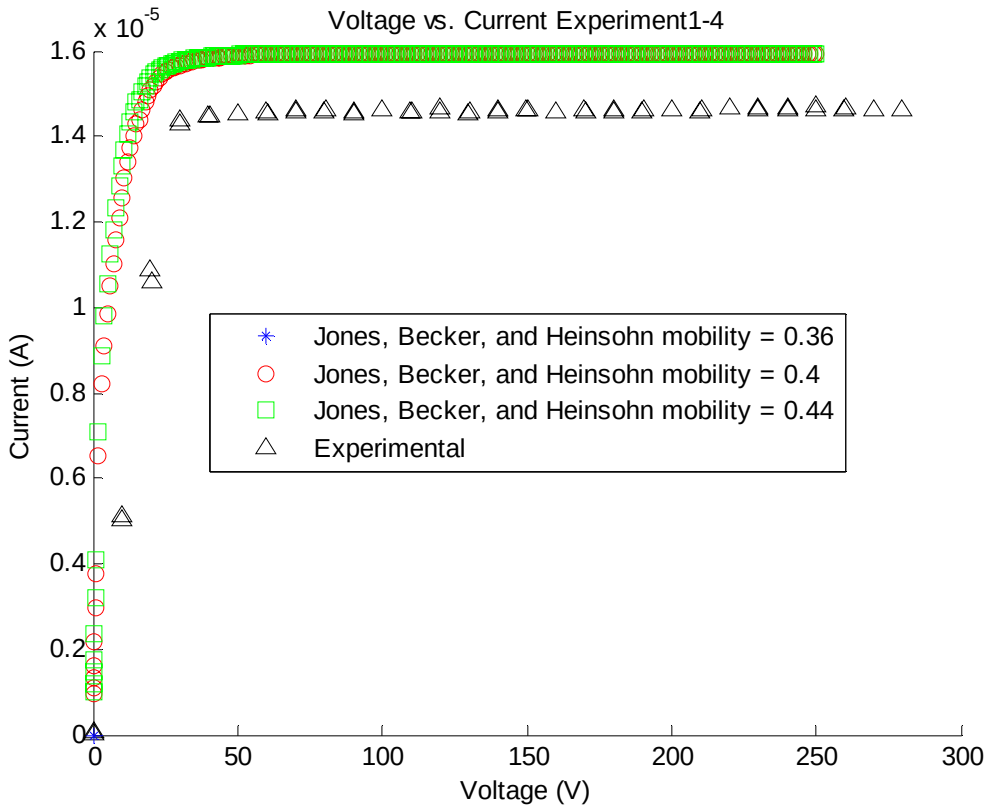

Figure E-22: Voltage vs. Current with Methane for Jones, Becker and Heinsohn Experiment 1-4

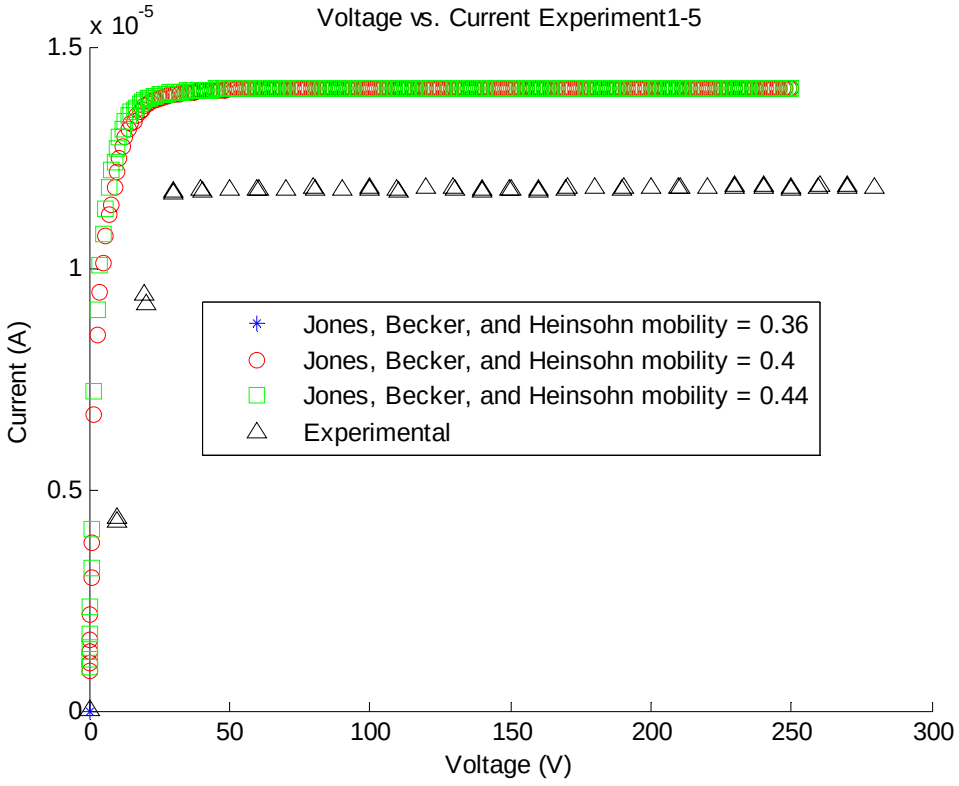

Figure E-23: Voltage vs. Current with Methane for Jones, Becker and Heinsohn Experiment 1-5

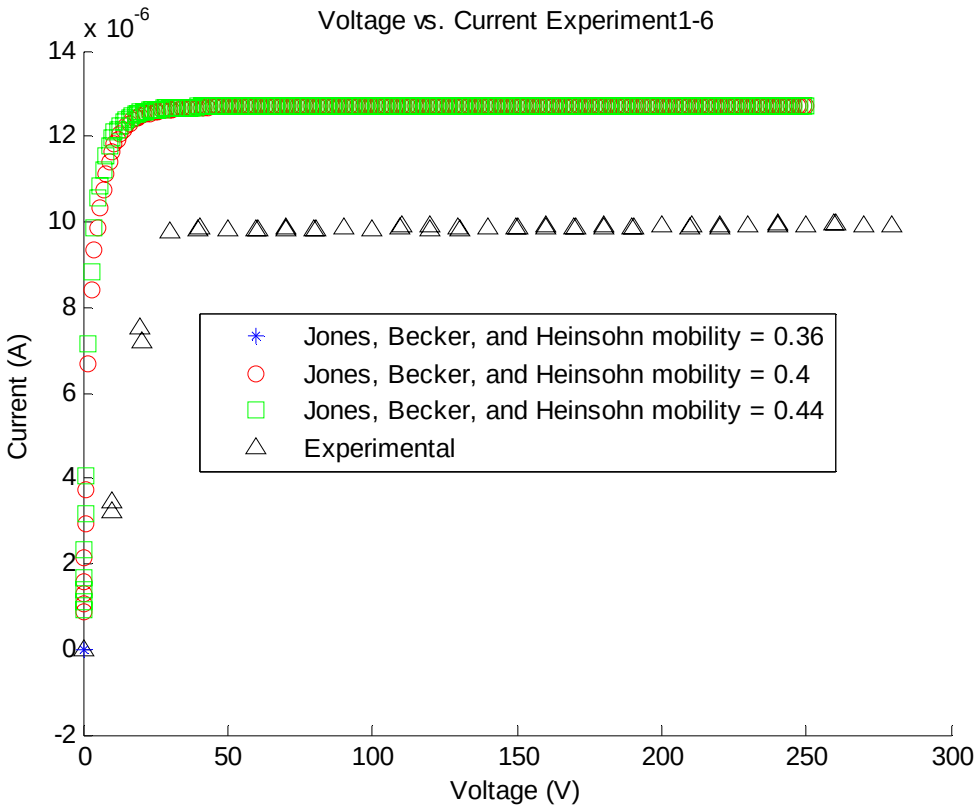

Figure E-24: Voltage vs. Current with Methane for Jones, Becker and Heinsohn Experiment 1-6

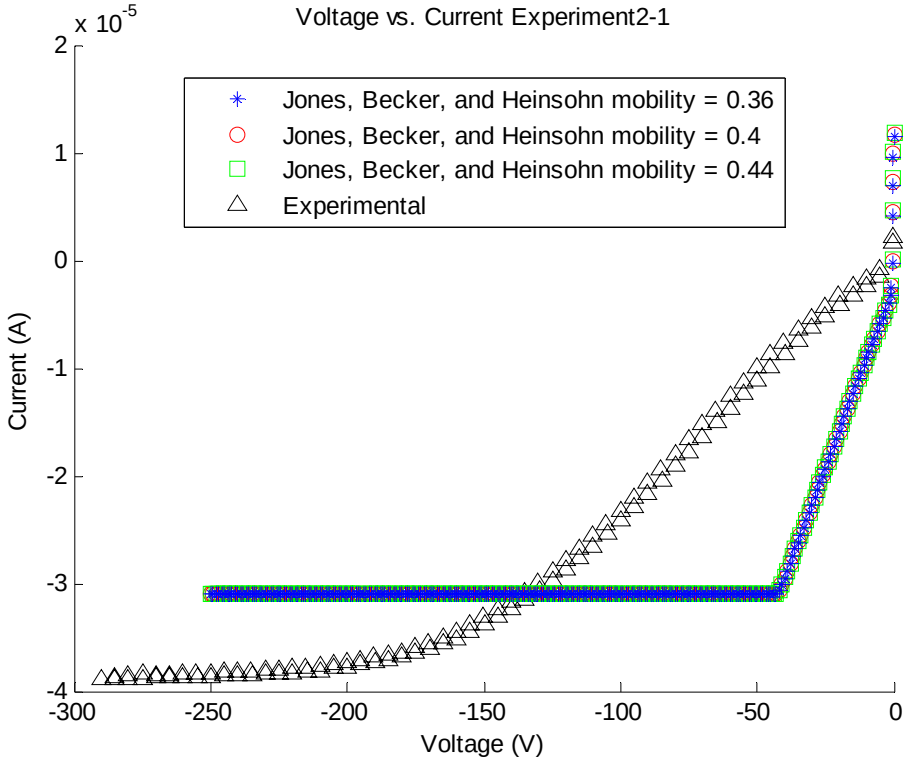

Figure E-25: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 2-1

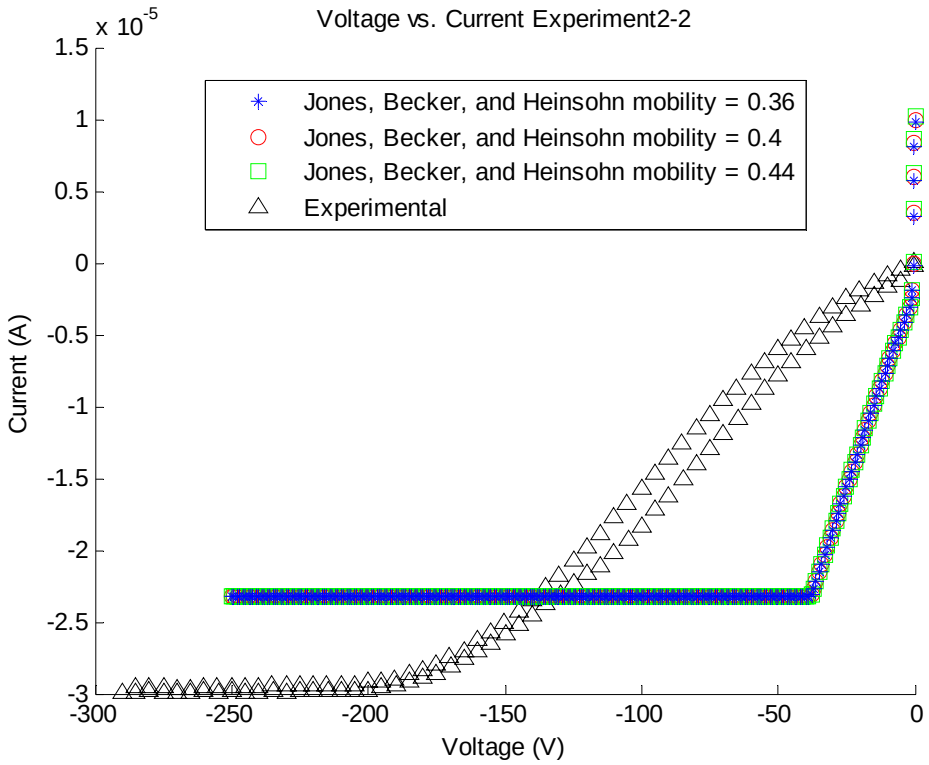

Figure E-26: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 2-2

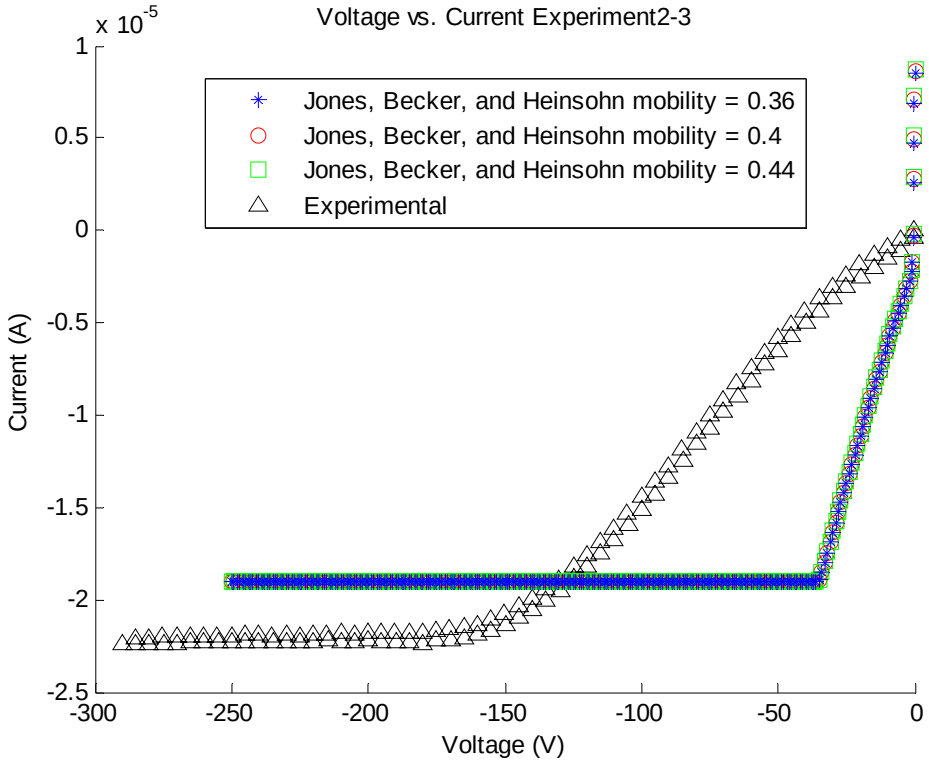

Figure E-27: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 2-3

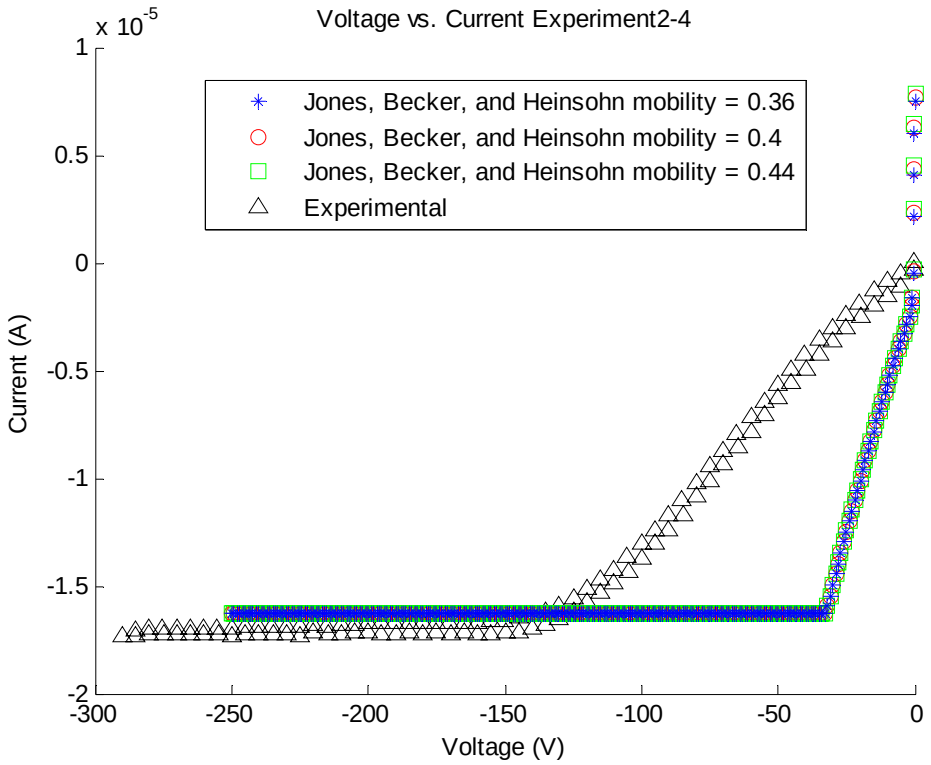

Figure E-28: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 2-4

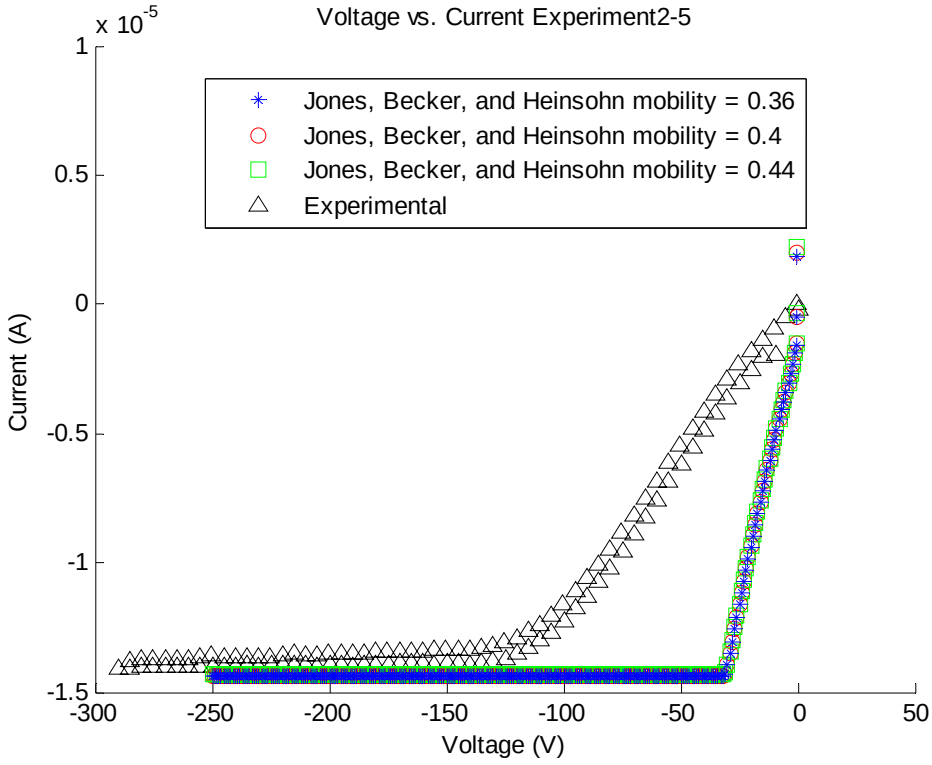

Figure E-29: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 2-5

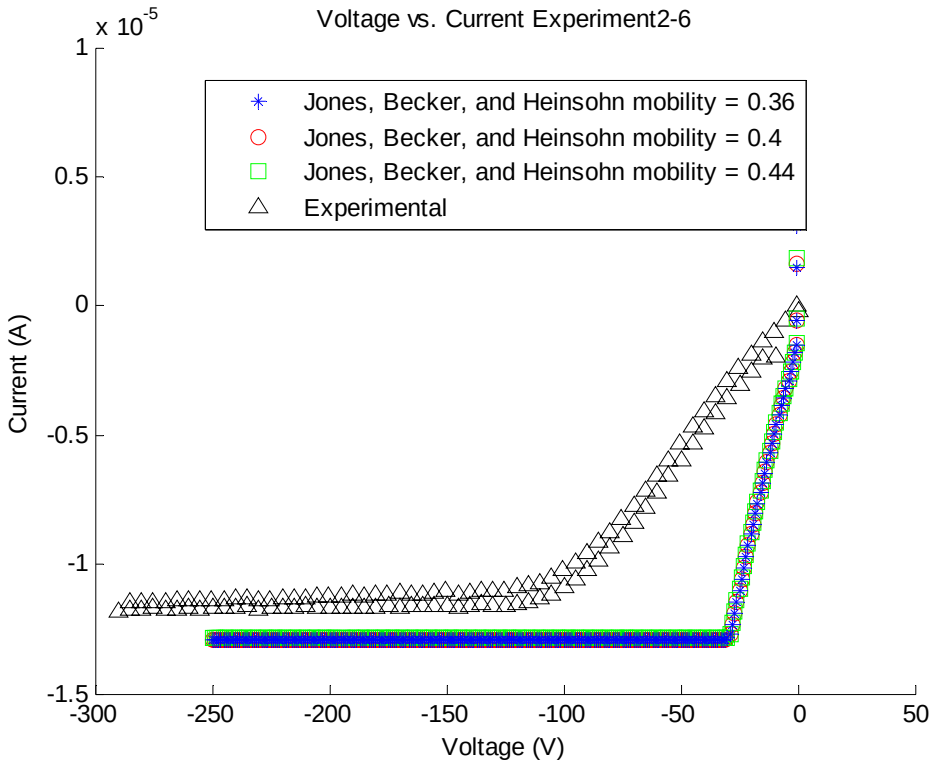

Figure E-30: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 2-6

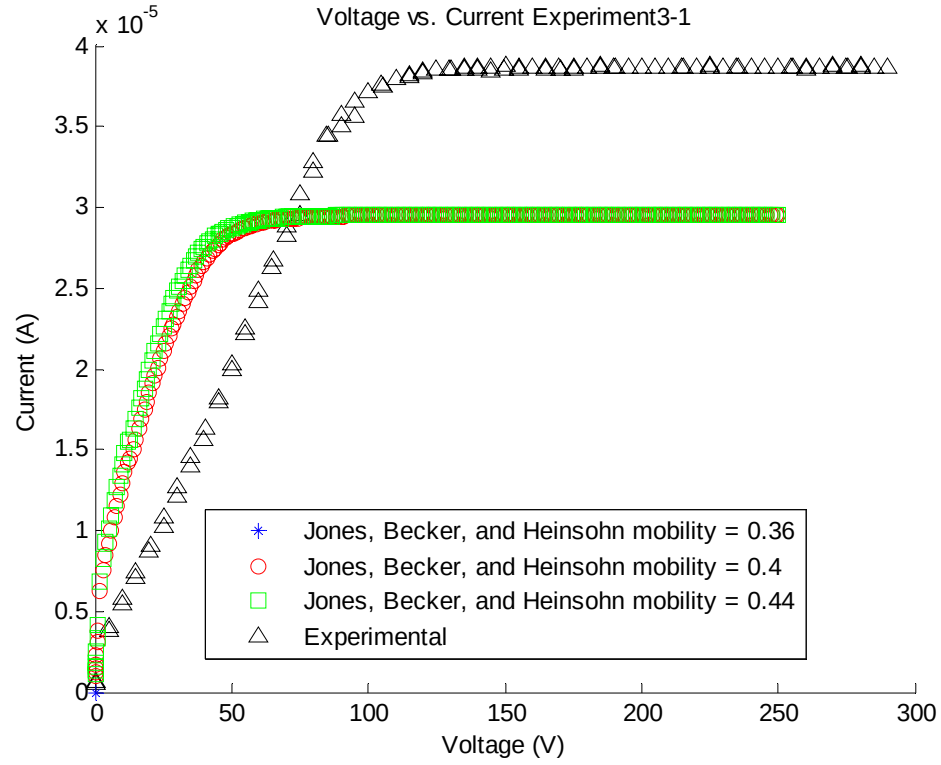

Figure E-31: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 3-1

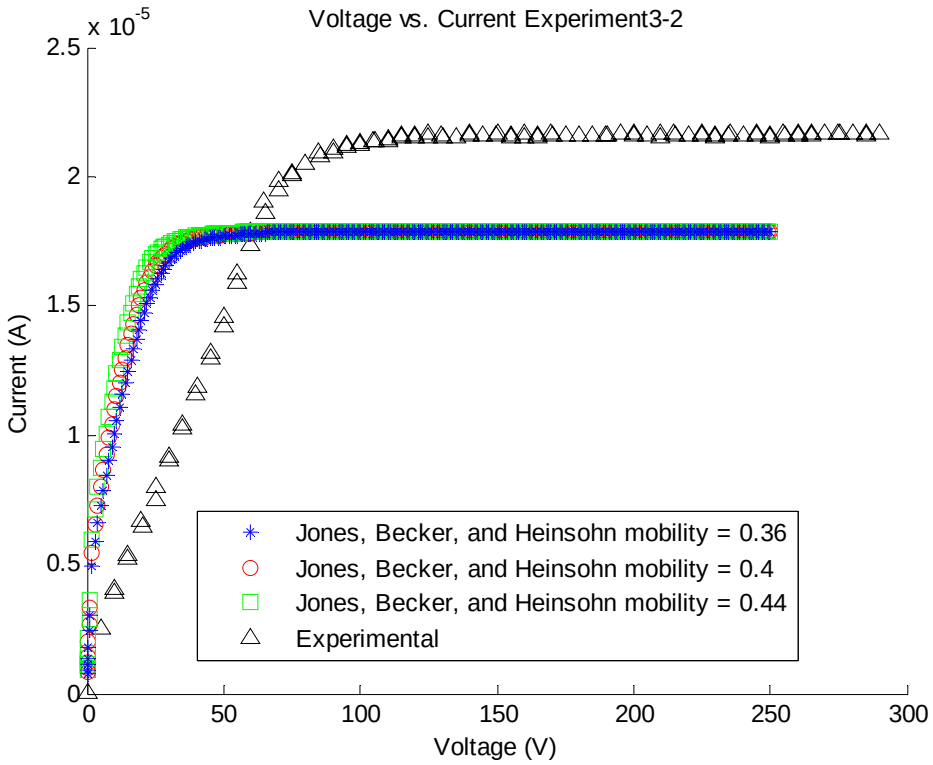

Figure E-32: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 3-2

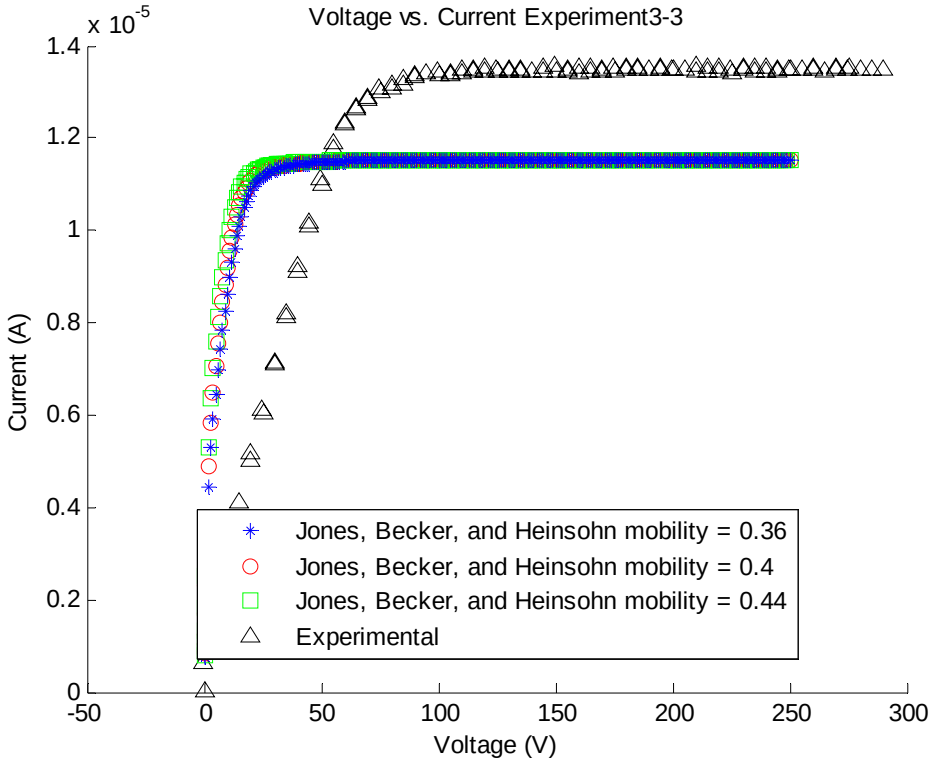

Figure E-33: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 3-3

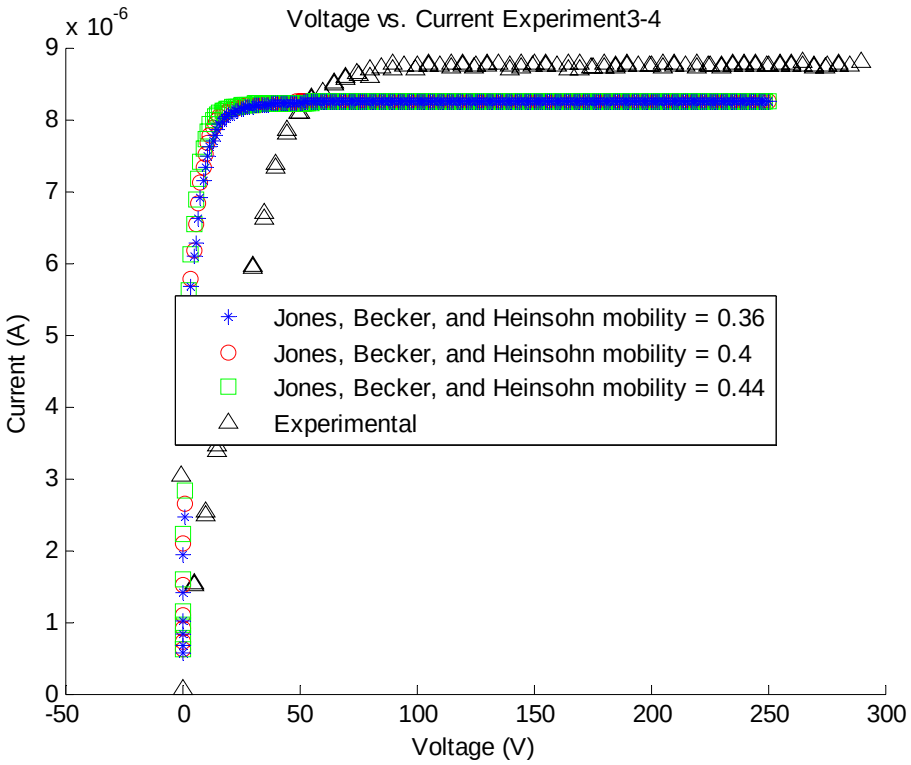

Figure E-34: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 3-4

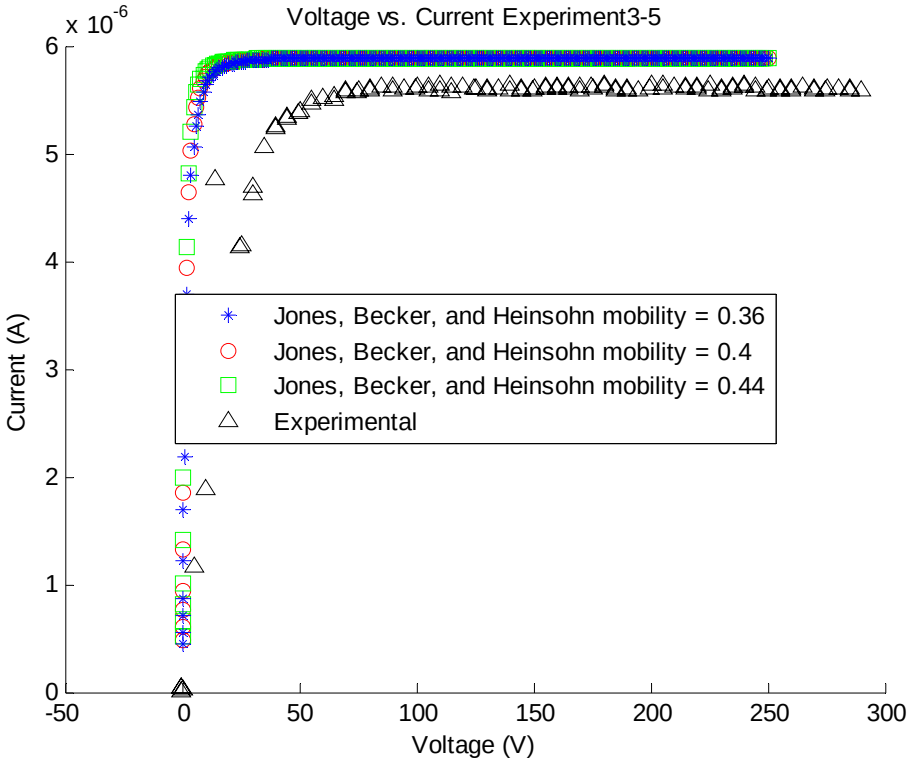

Figure E-35: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 3-5

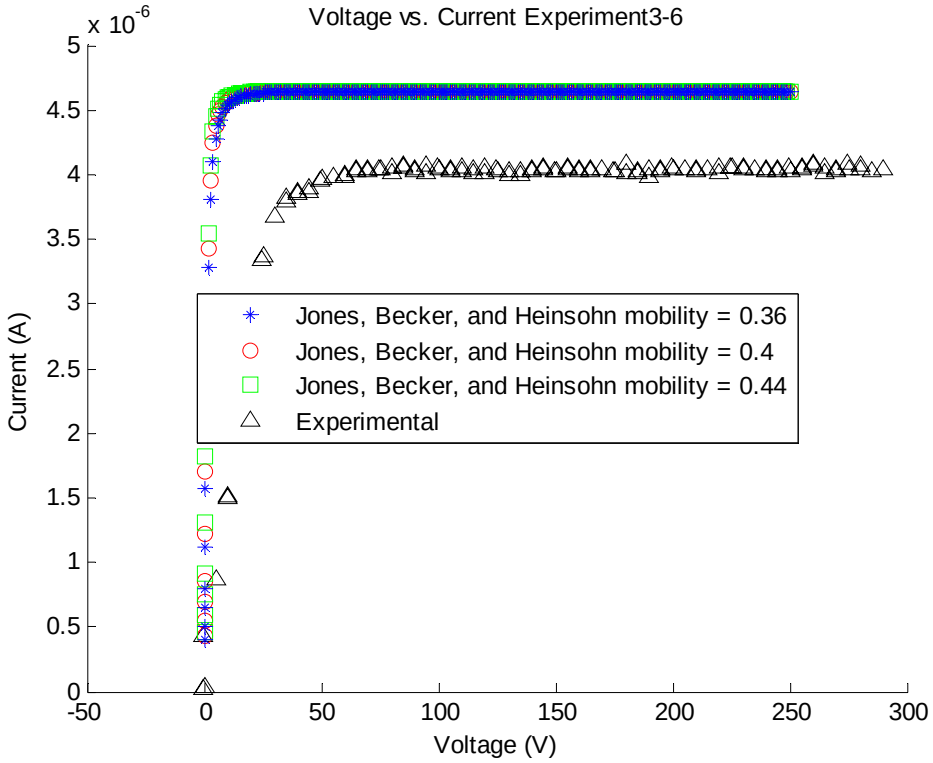

Figure E-36: Voltage vs. Current with Methane for Jones, Becker, and Heinsohn Experiment 3-6

E-3: V-I Plots of Methane Combustion Using Pederson and Brown

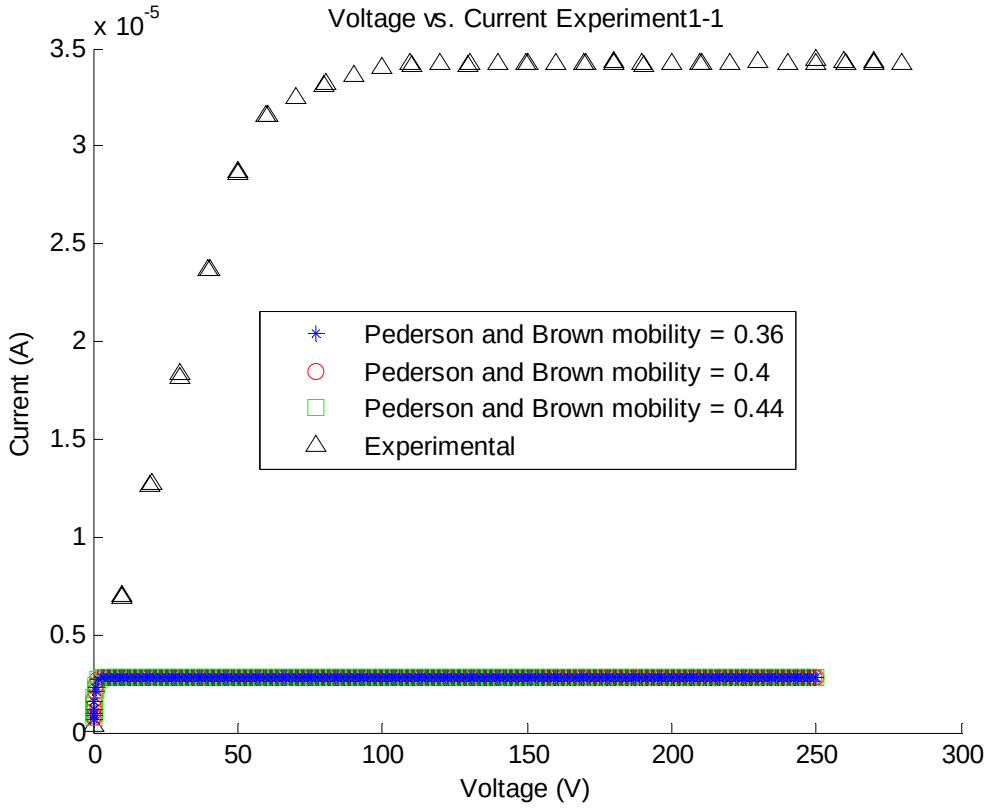

Figure E-37: Voltage vs. Current with Methane for Pederson and Brown Experiment 1-1

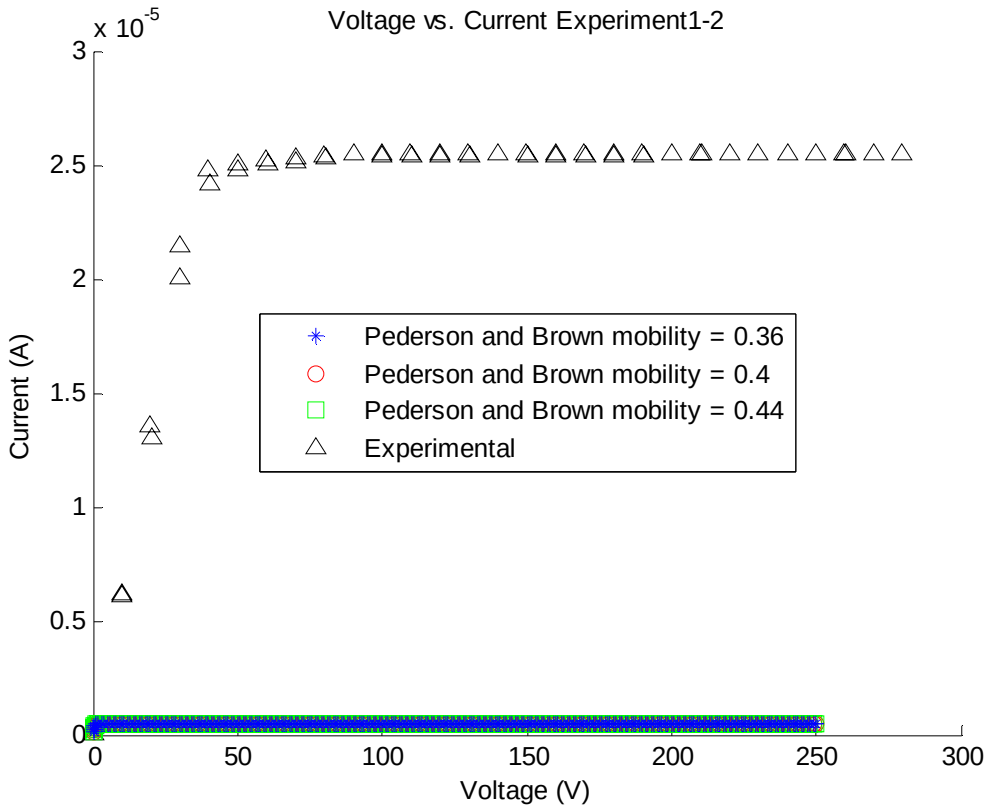

Figure E-38: Voltage vs. Current with Methane for Pederson and Brown Experiment 1-2

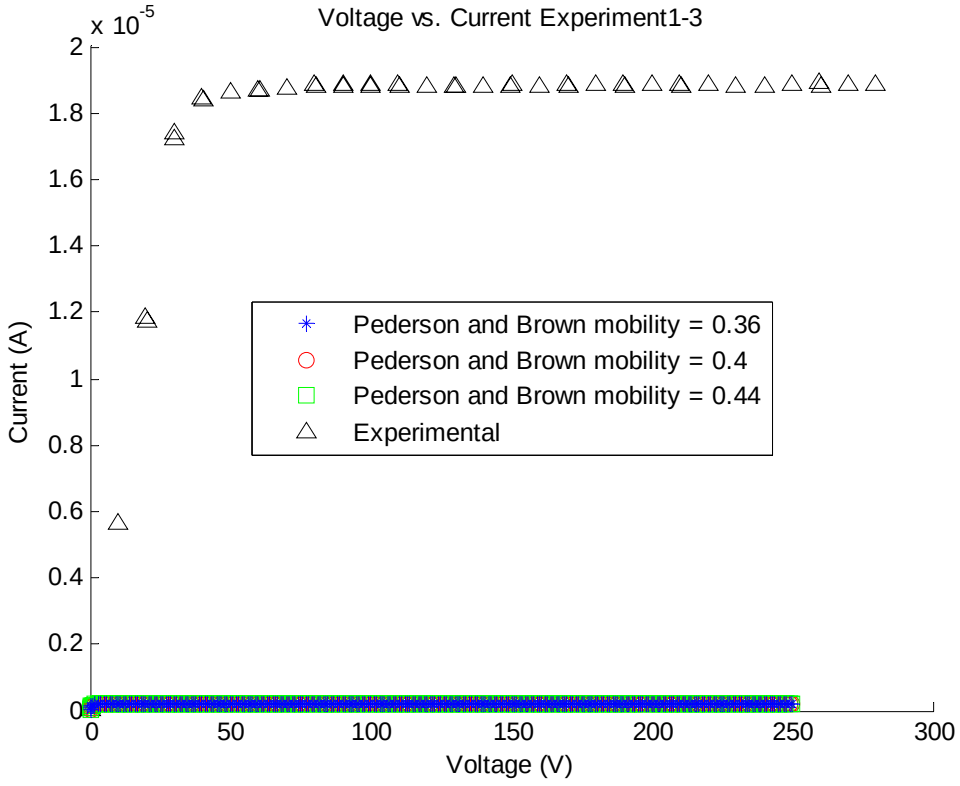

Figure E-39: Voltage vs. Current with Methane for Pederson and Brown Experiment 1-3

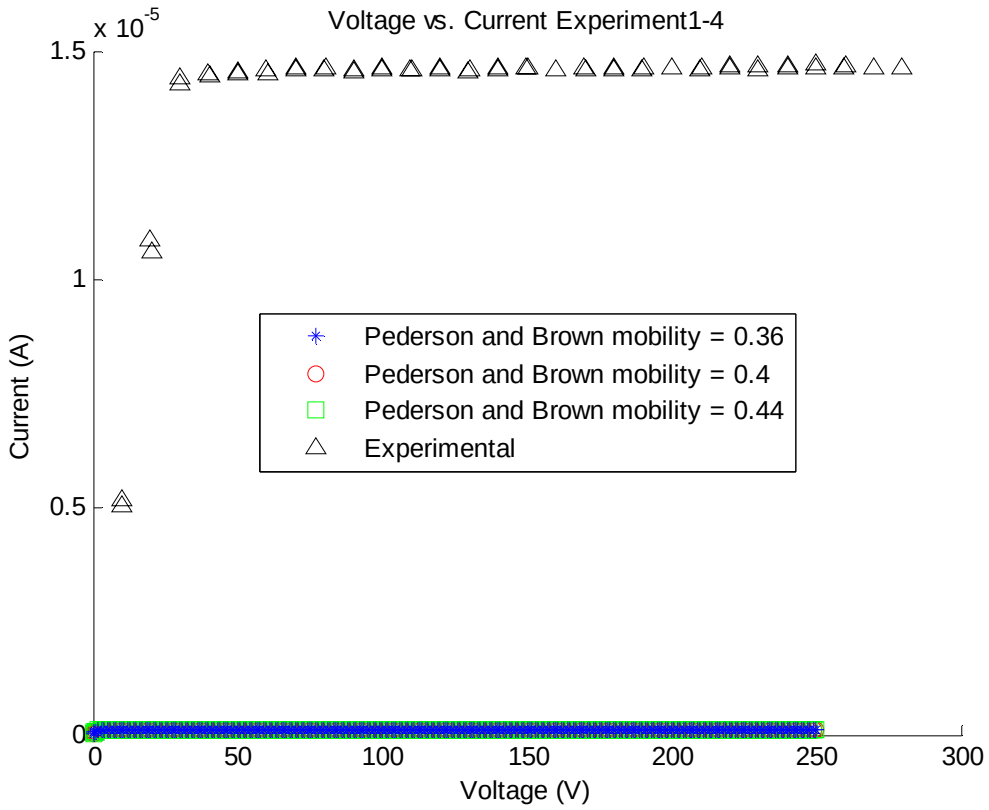

Figure E-40: Voltage vs. Current with Methane for Pederson and Brown Experiment 1-4

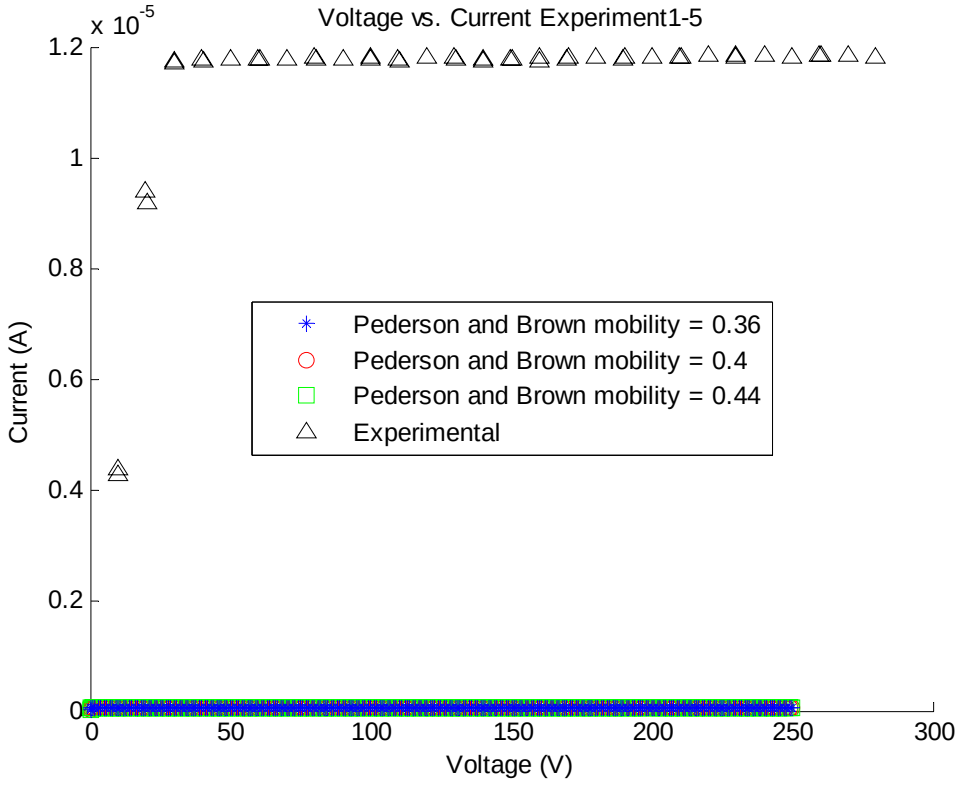

Figure E-41: Voltage vs. Current with Methane for Pederson and Brown Experiment 1-5

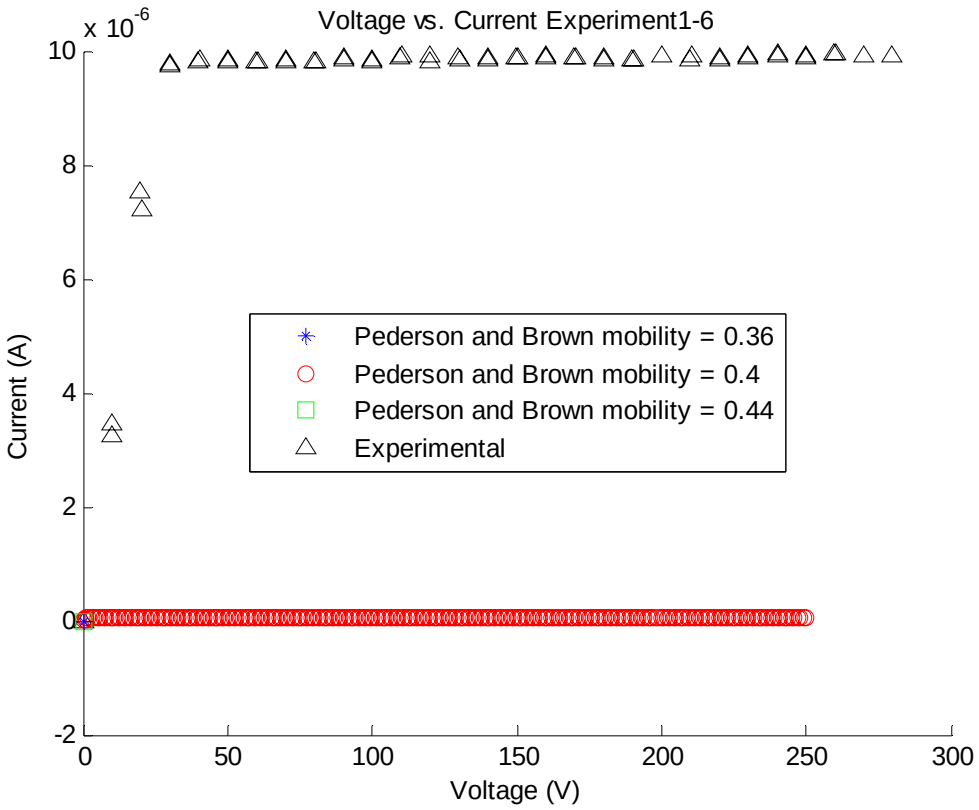

Figure E-42: Voltage vs. Current with Methane for Pederson and Brown Experiment 1-6

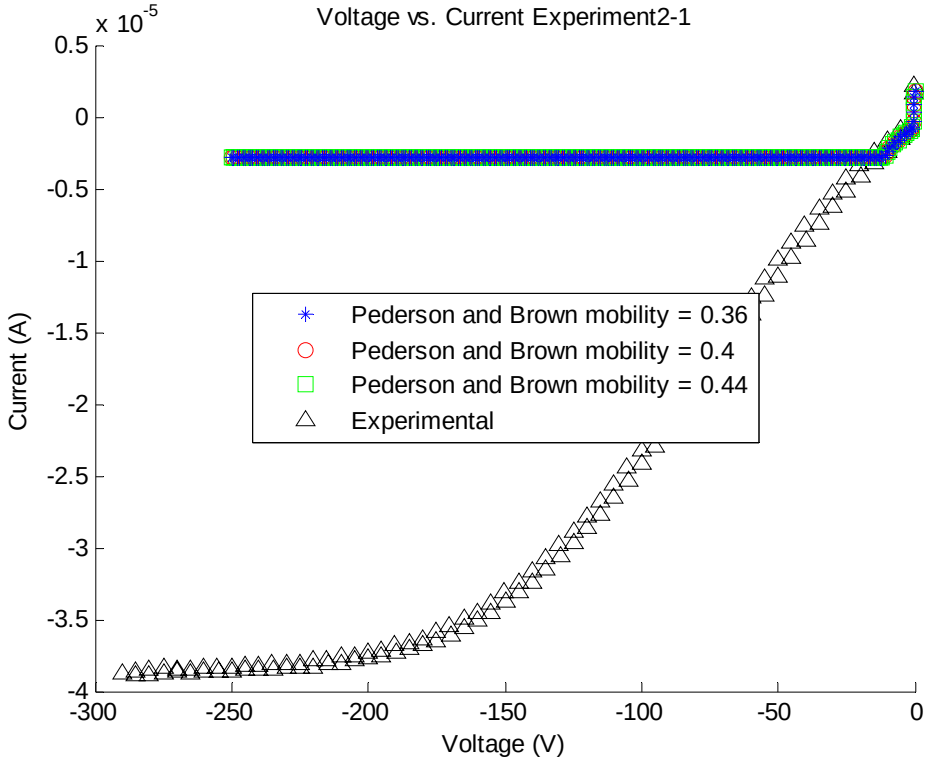

Figure E-43: Voltage vs. Current with Methane for Pederson and Brown Experiment 2-1

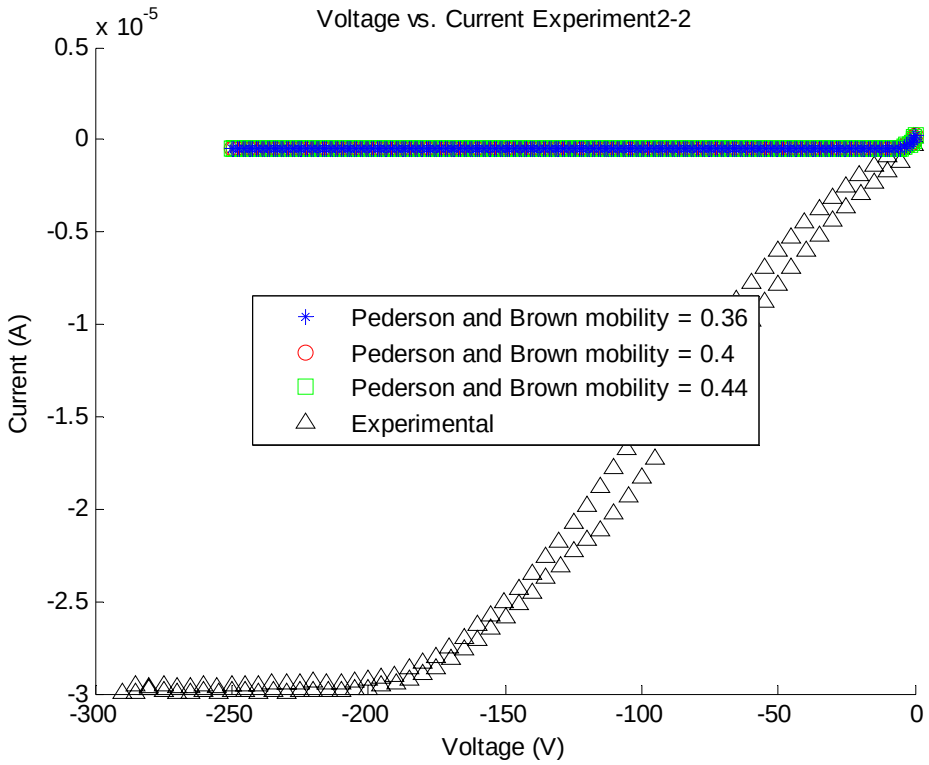

Figure E-44: Voltage vs. Current with Methane for Pederson and Brown Experiment 2-2

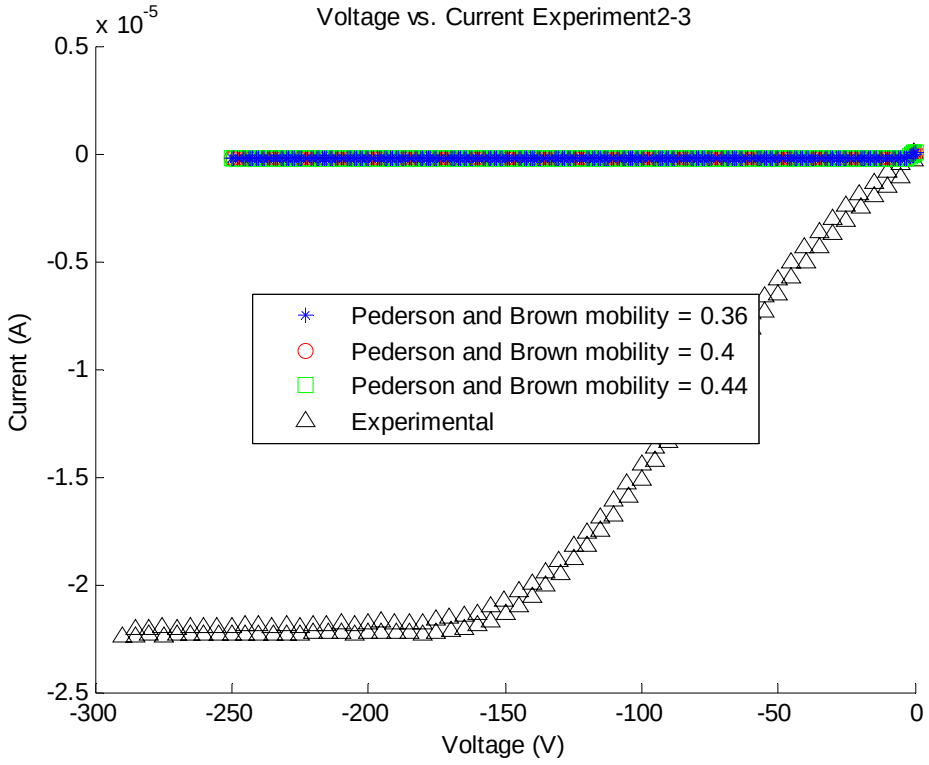

Figure E-45: Voltage vs. Current with Methane for Pederson and Brown Experiment 2-3

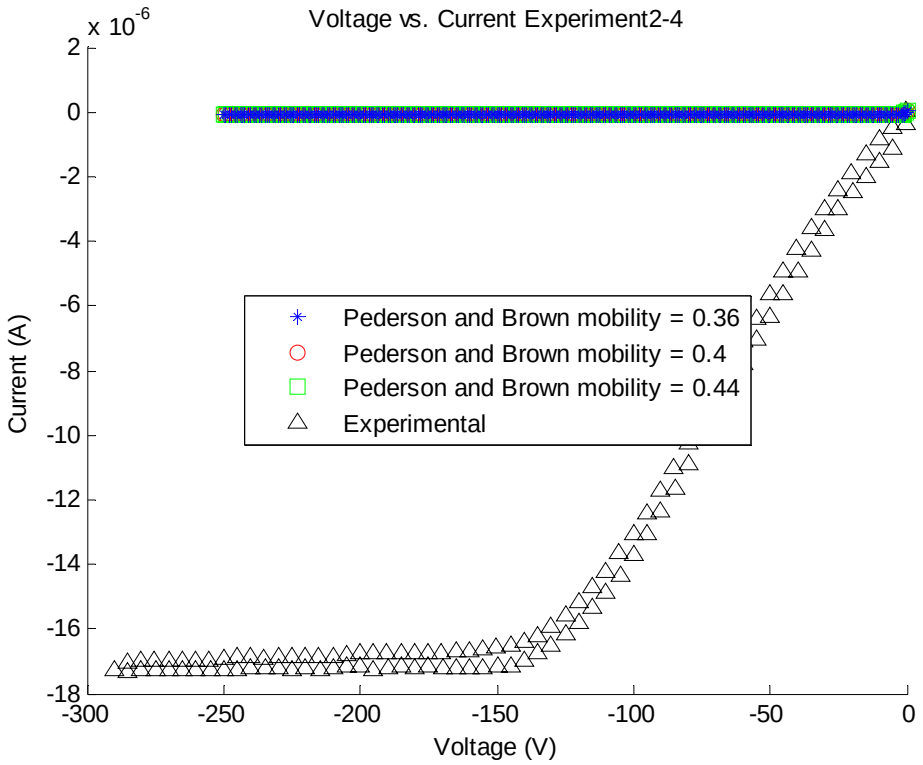

Figure E-46: Voltage vs. Current with Methane for Pederson and Brown Experiment 2-4

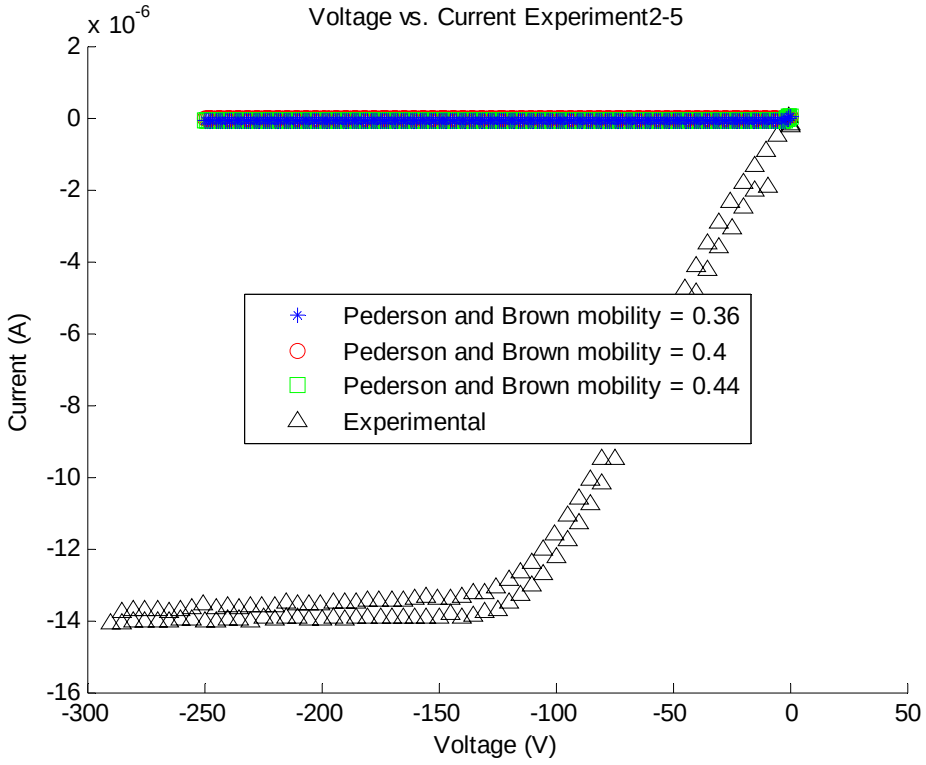

Figure E-47: Voltage vs. Current with Methane for Pederson and Brown Experiment 2-5

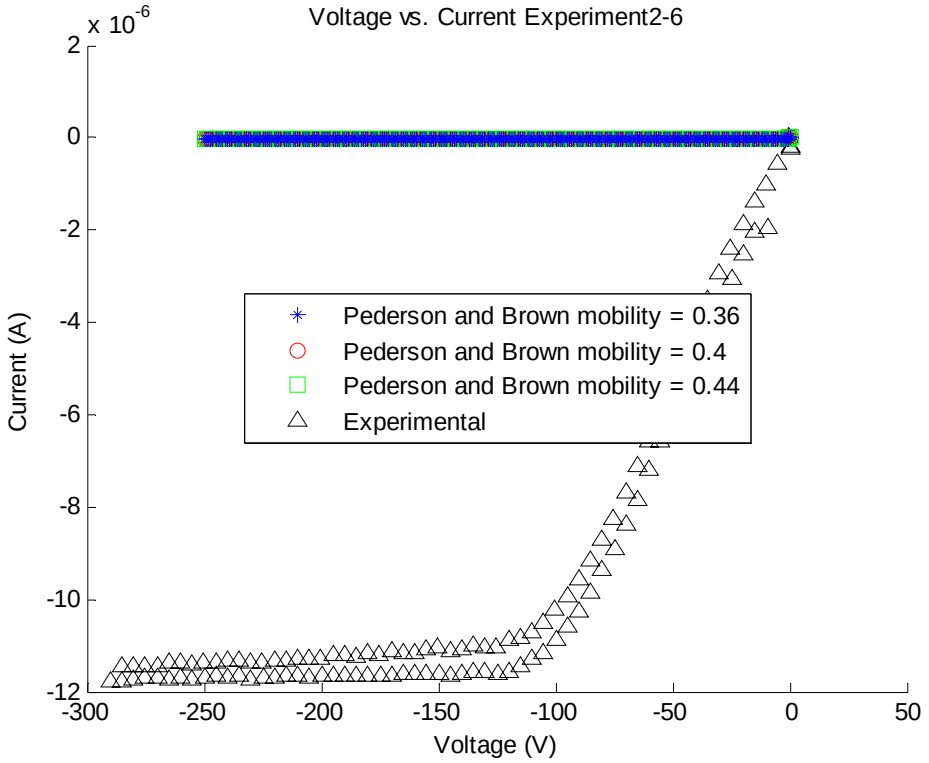

Figure E-48: Voltage vs. Current with Methane for Pederson and Brown Experiment 2-6

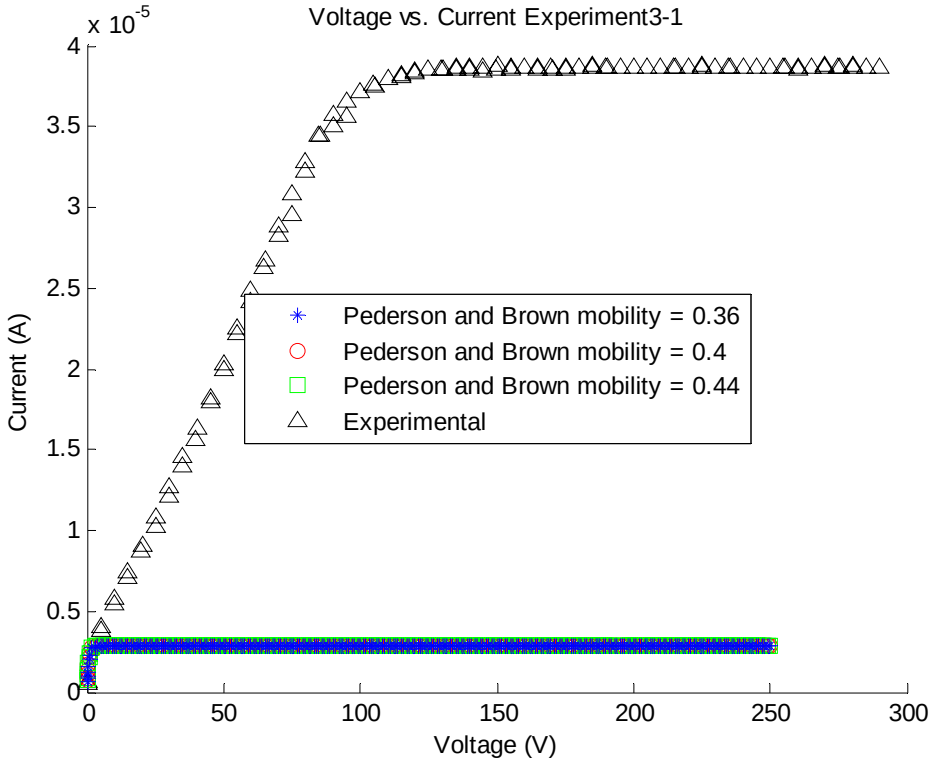

Figure E-49: Voltage vs. Current with Methane for Pederson and Brown Experiment 3-1

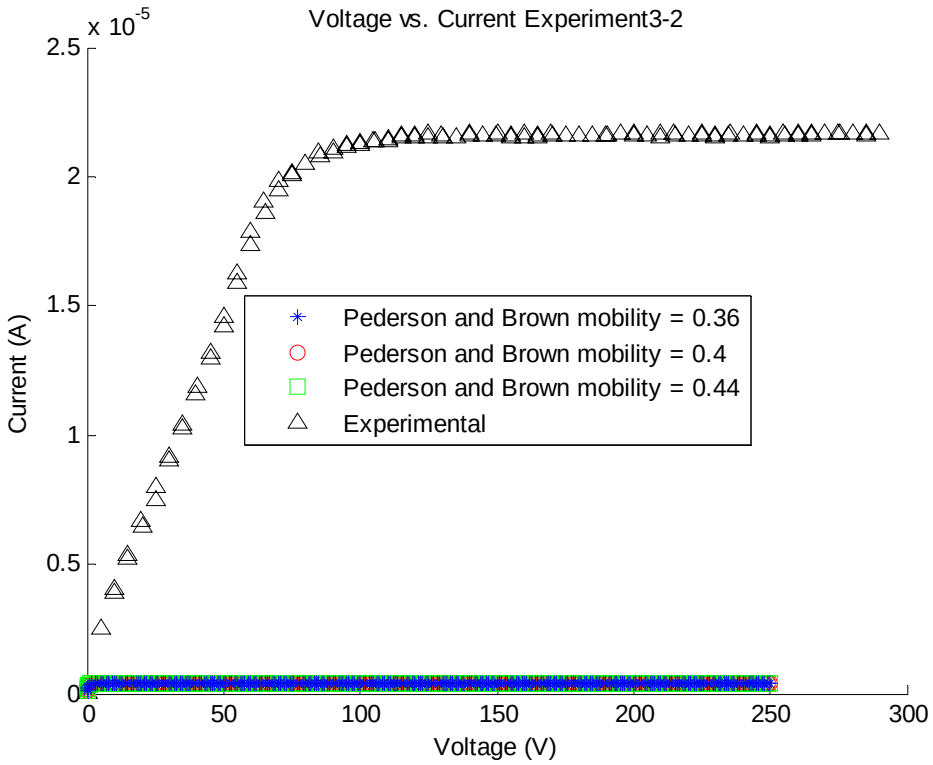

Figure E-50: Voltage vs. Current with Methane for Pederson and Brown Experiment 3-2

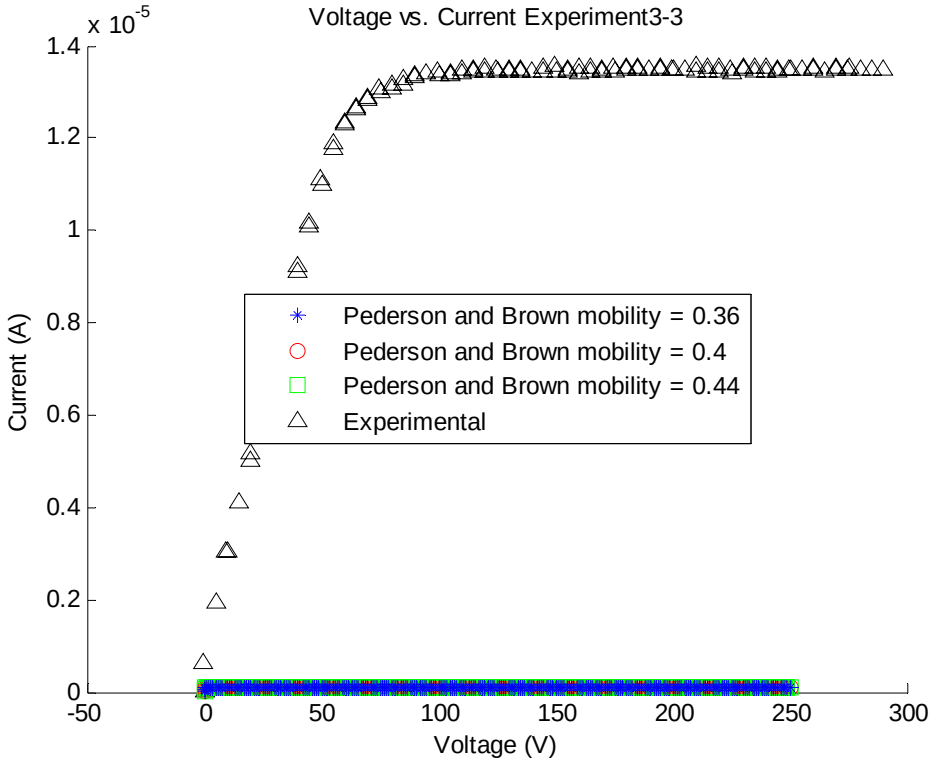

Figure E-51: Voltage vs. Current with Methane for Pederson and Brown Experiment 3-3

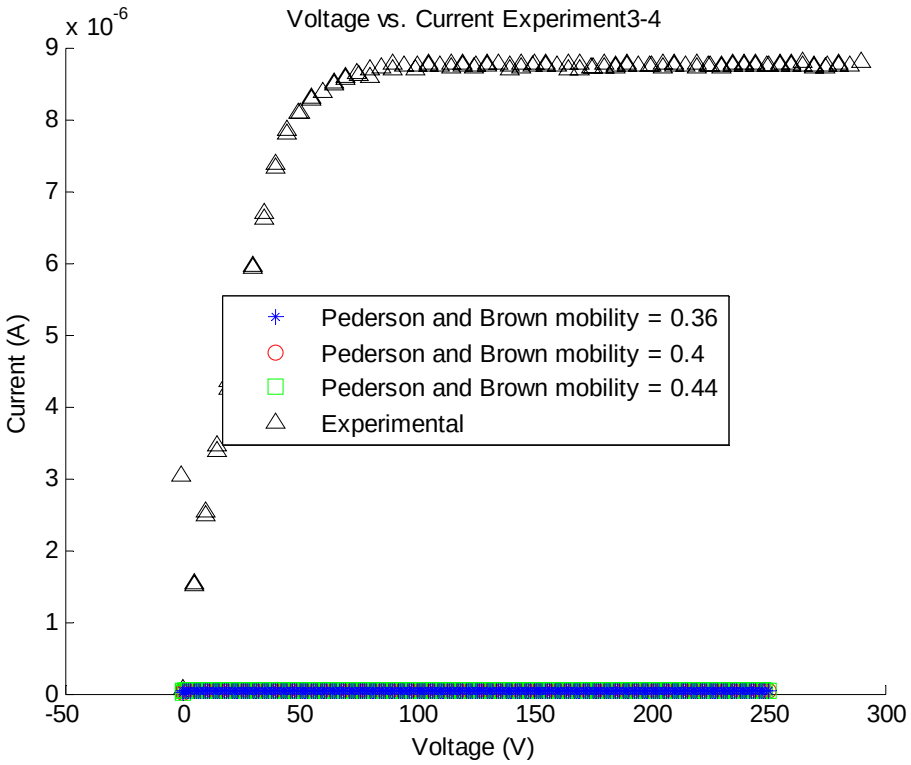

Figure E-52: Voltage vs. Current with Methane for Pederson and Brown Experiment 3-4

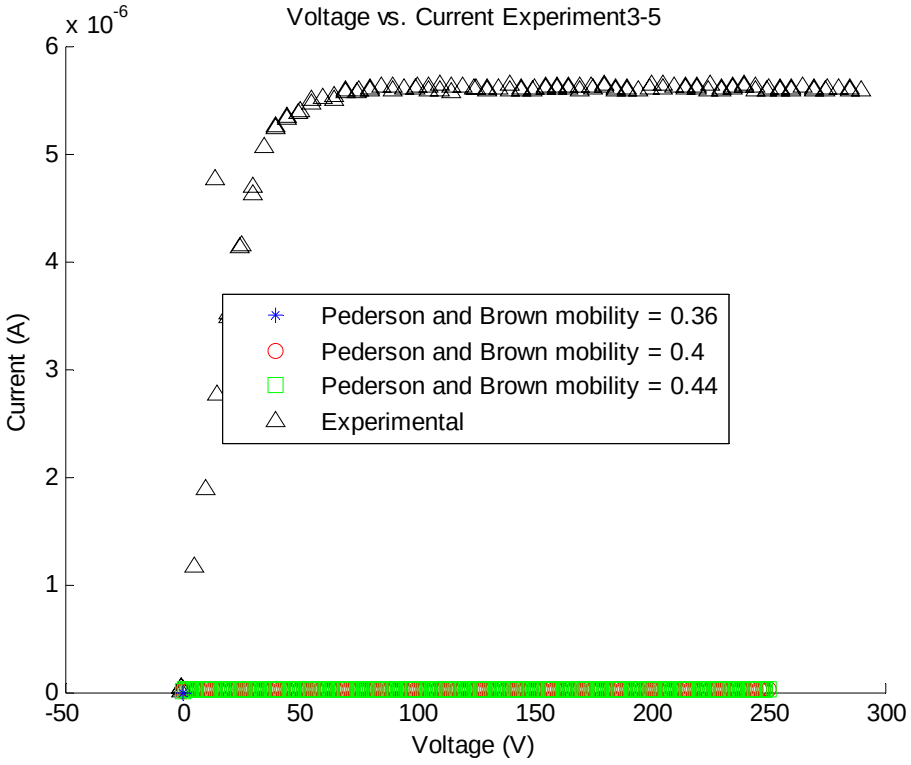

Figure E-53: Voltage vs. Current with Methane for Pederson and Brown Experiment 3-5

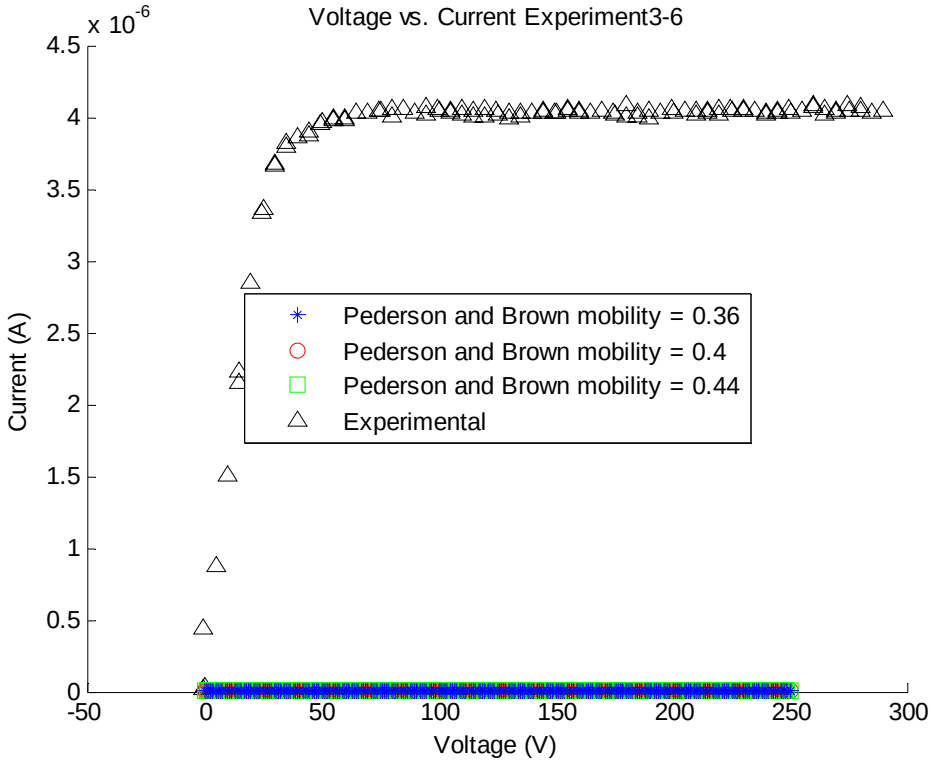

Figure E-54: Voltage vs. Current with Methane for Pederson and Brown Experiment 3-6

E-4: V-I Plots of Methane Combustion Using Peters

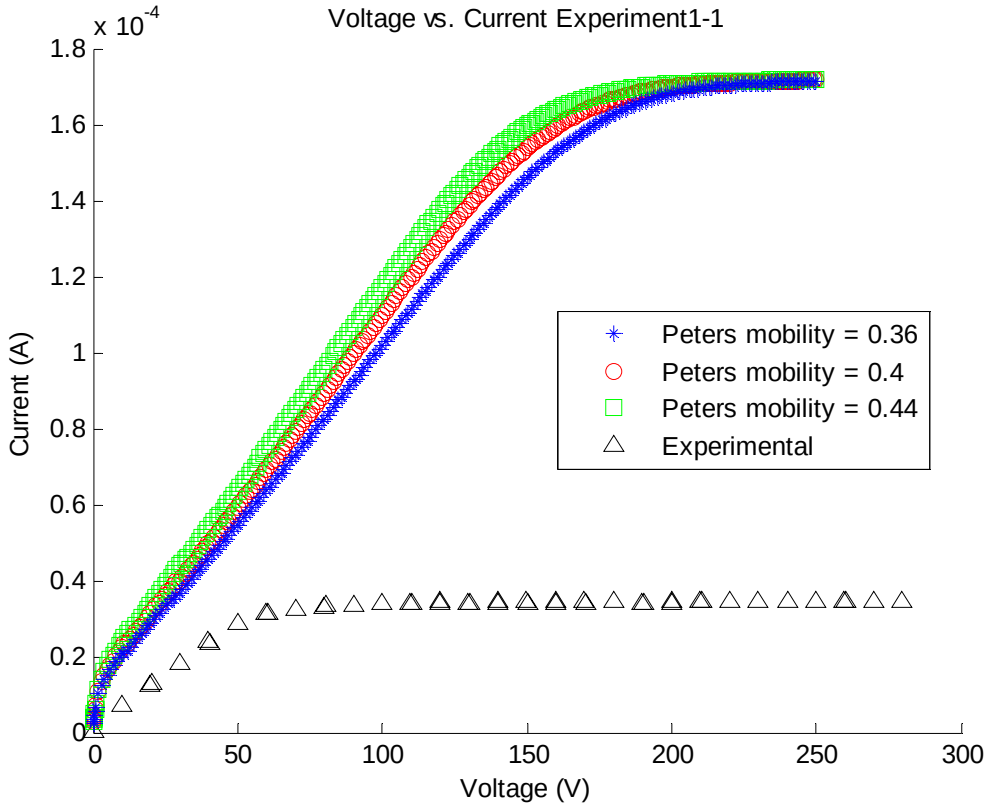

Figure E-55: Voltage vs. Current with Methane for Peters Experiment 1-1

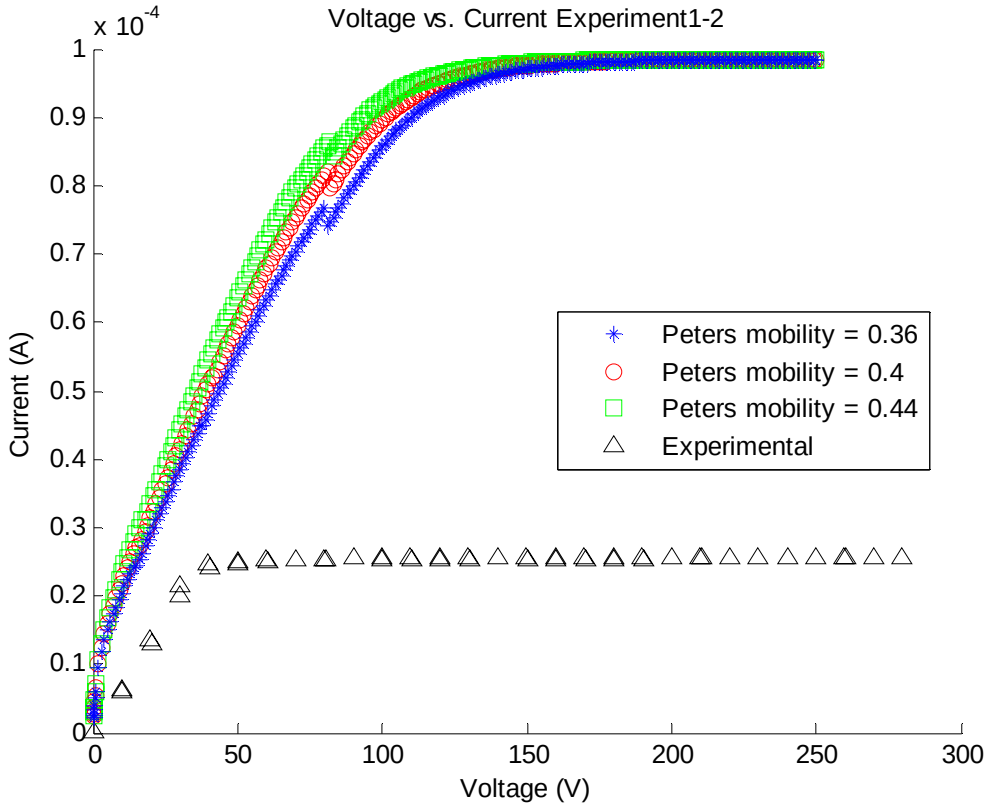

Figure E-56: Voltage vs. Current with Methane for Peters Experiment 1-2

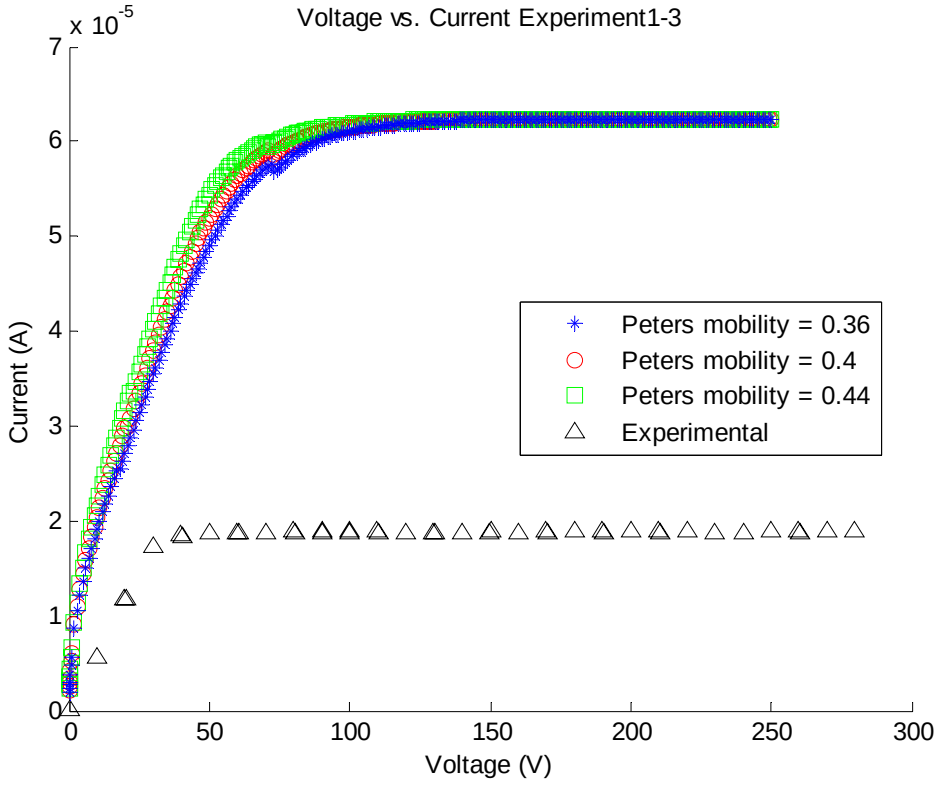

Figure E-57: Voltage vs. Current with Methane for Peters Experiment 1-3

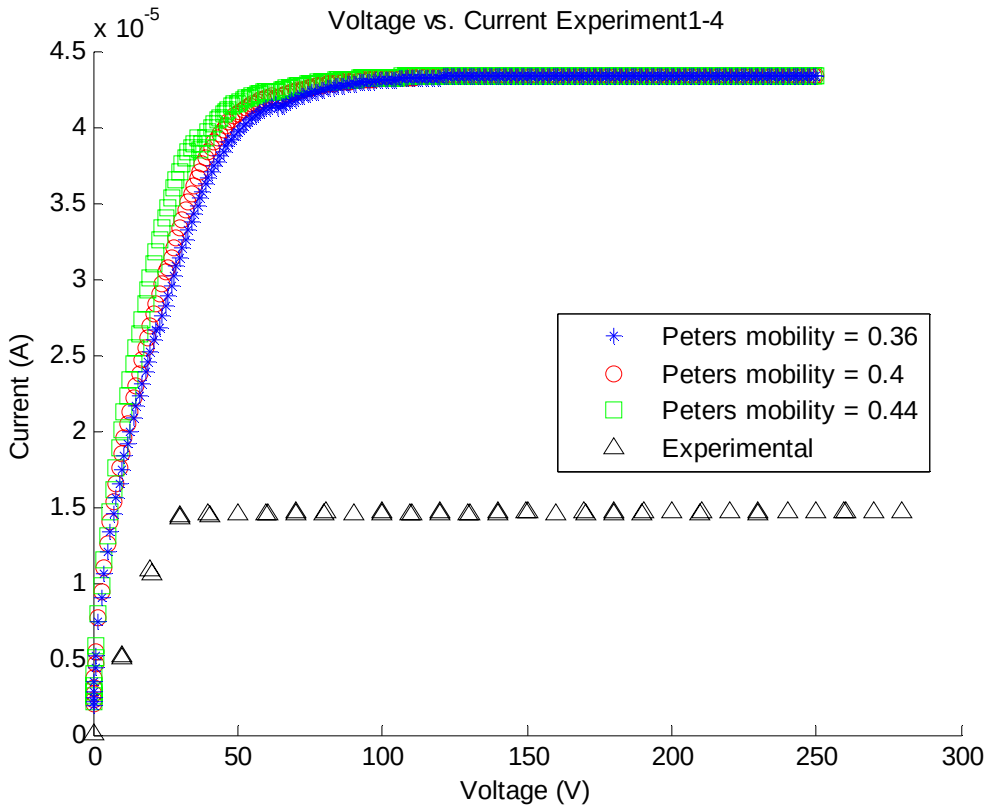

Figure E-58: Voltage vs. Current with Methane for Peters Experiment 1-4

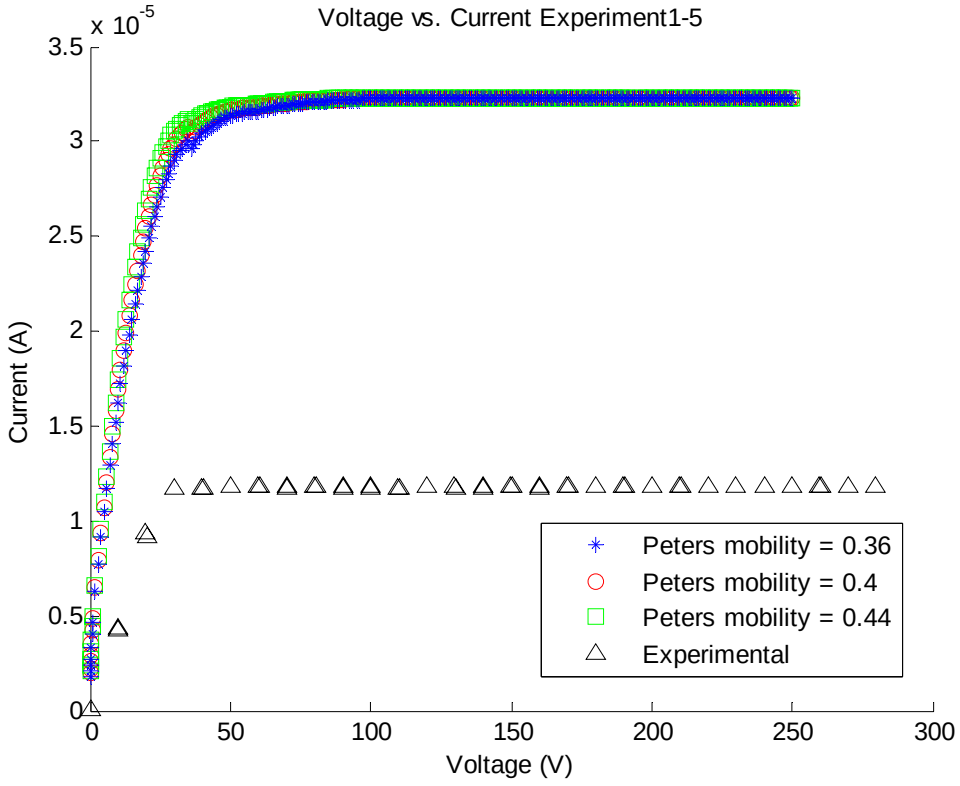

Figure E-59: Voltage vs. Current with Methane for Peters Experiment 1-5

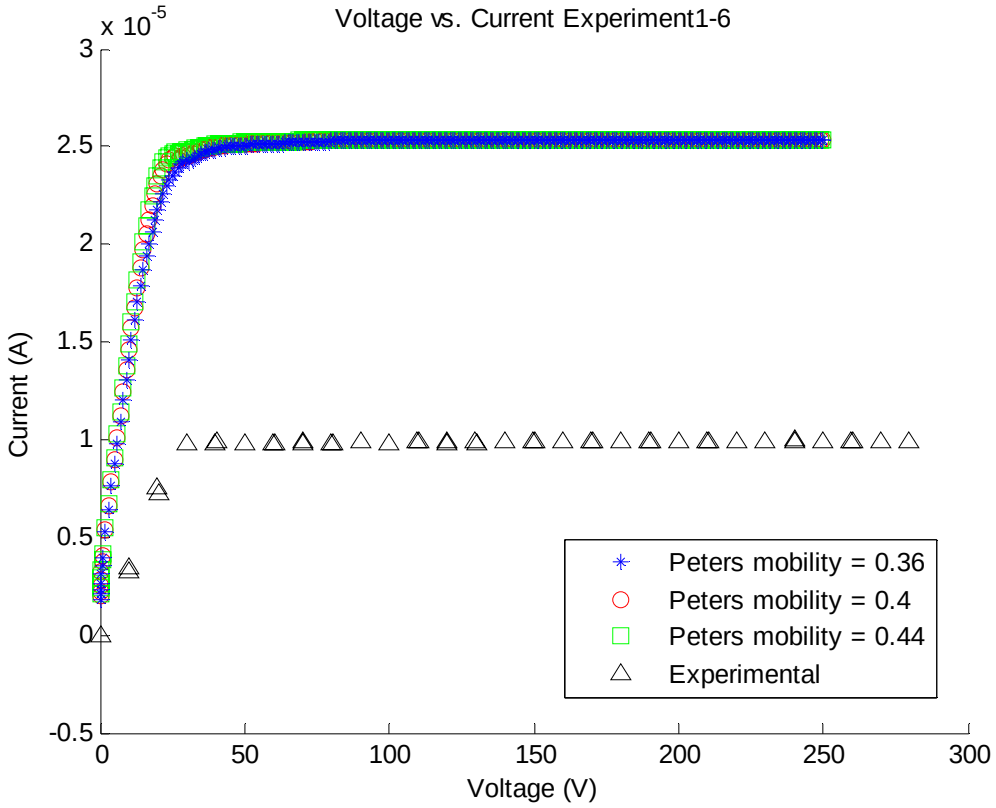

Figure E-60: Voltage vs. Current with Methane for Peters Experiment 1-6

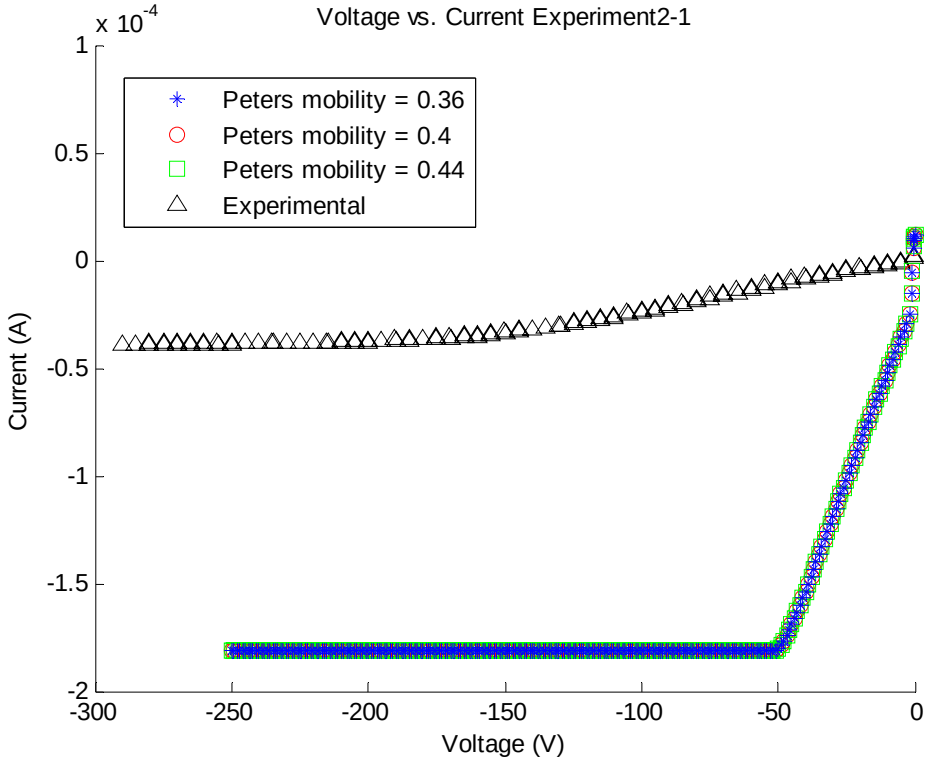

Figure E-61: Voltage vs. Current with Methane for Peters Experiment 2-1

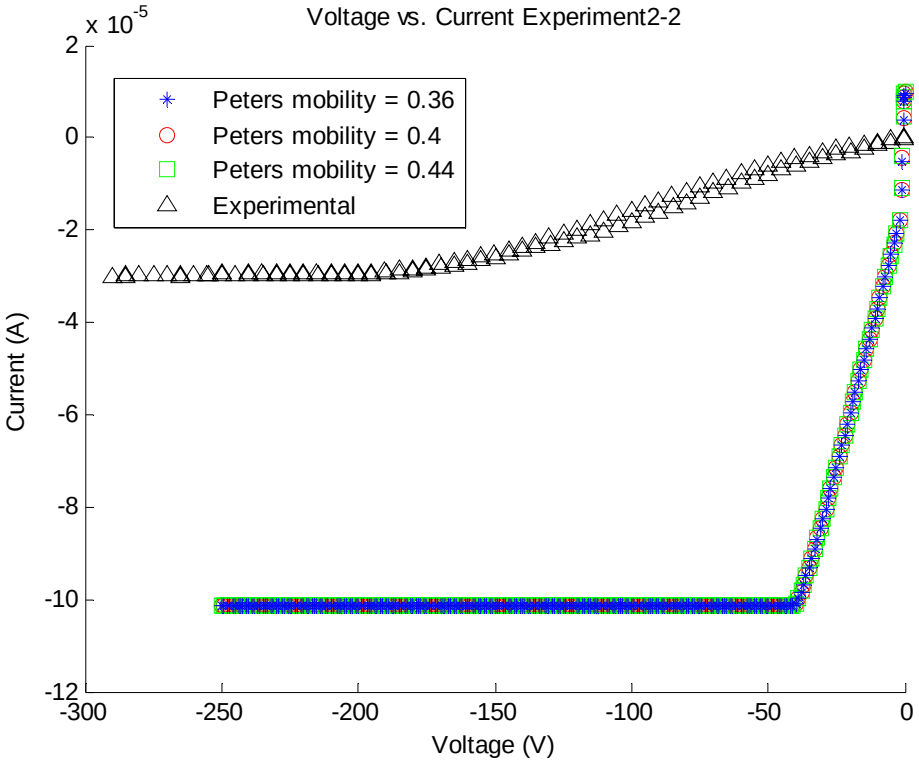

Figure E-62: Voltage vs. Current with Methane for Peters Experiment 2-2

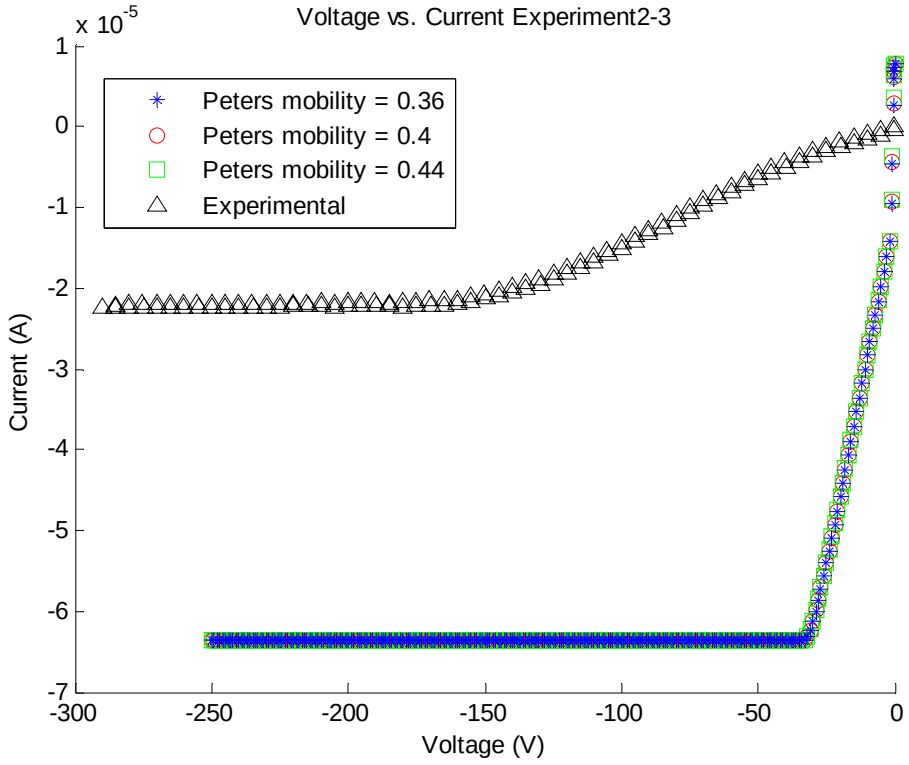

Figure E-63: Voltage vs. Current with Methane for Peters Experiment 2-3

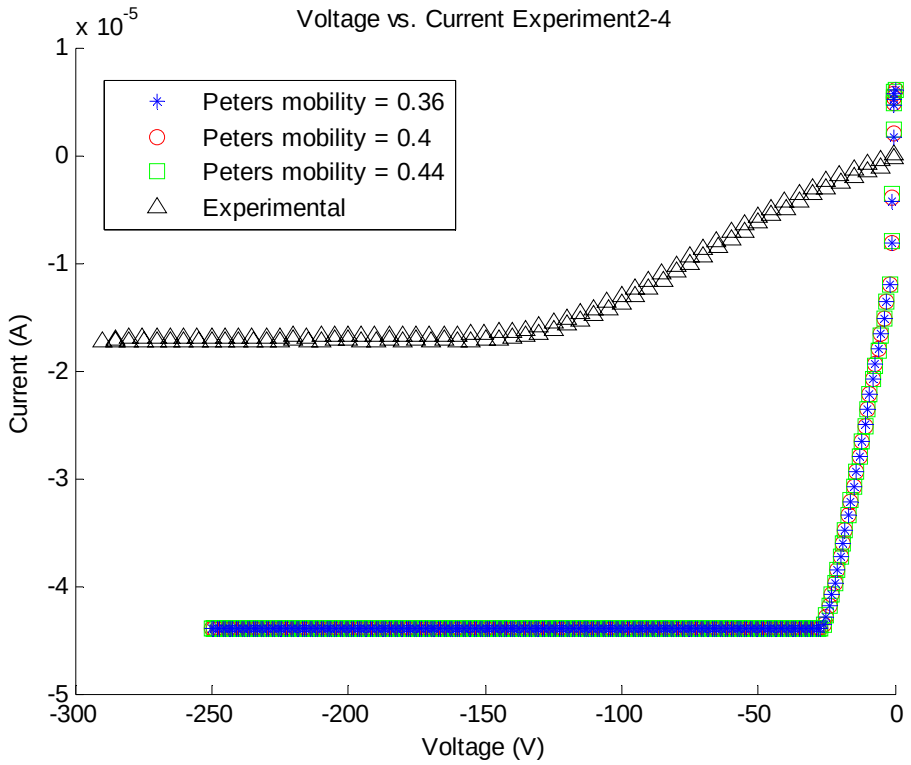

Figure E-64: Voltage vs. Current with Methane for Peters Experiment 2-4

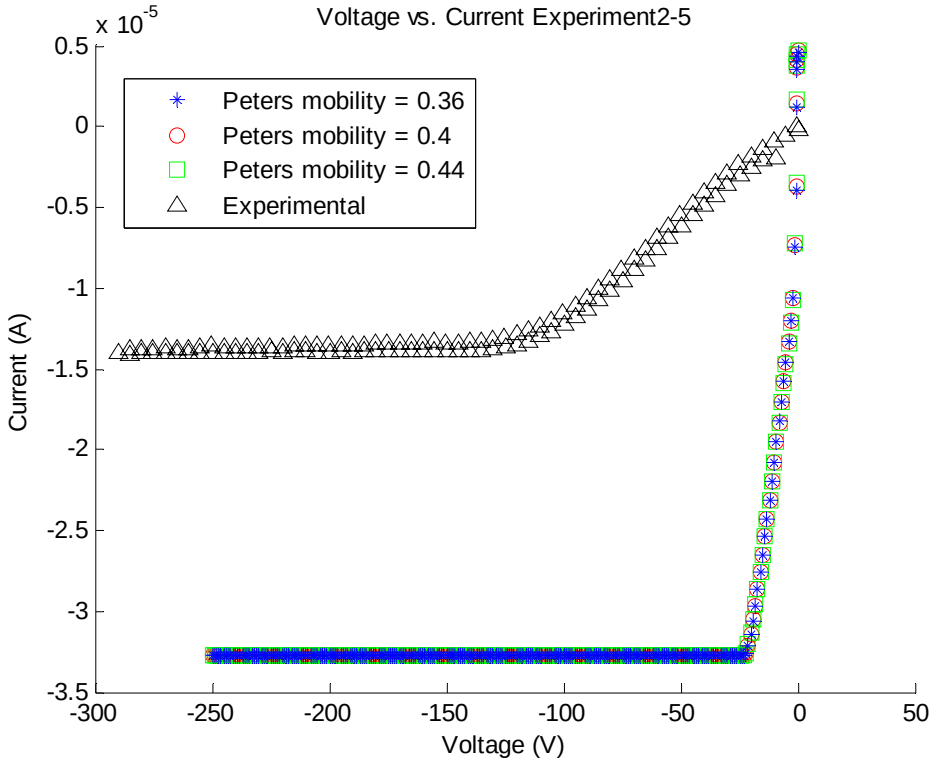

Figure E-65: Voltage vs. Current with Methane for Peters Experiment 2-5

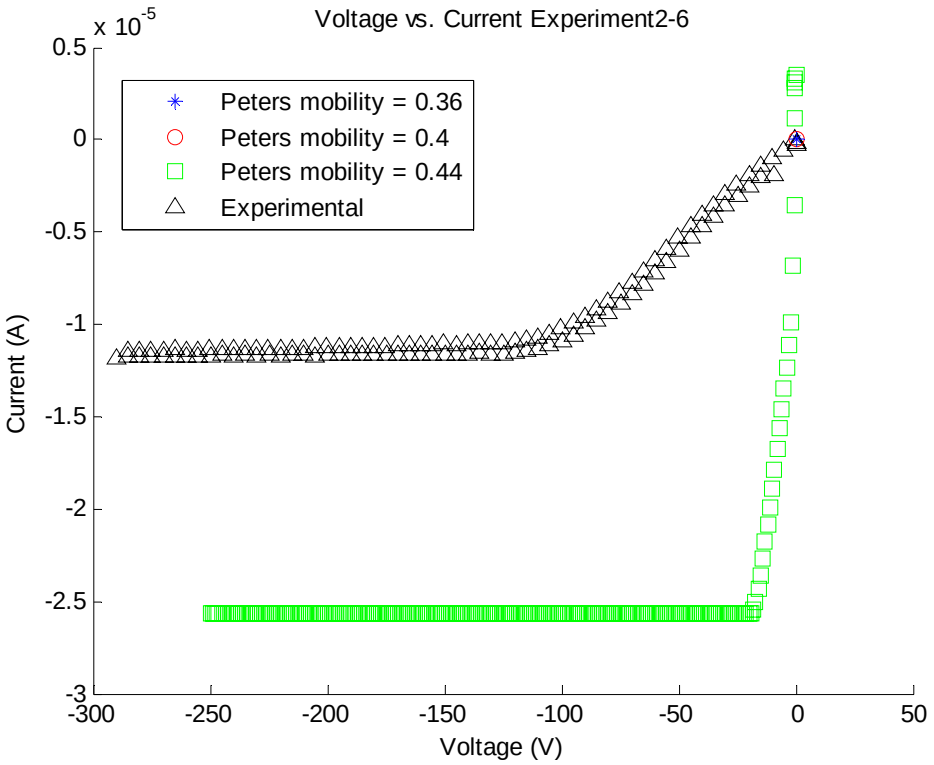

Figure E-66: Voltage vs. Current with Methane for Peters Experiment 2-6

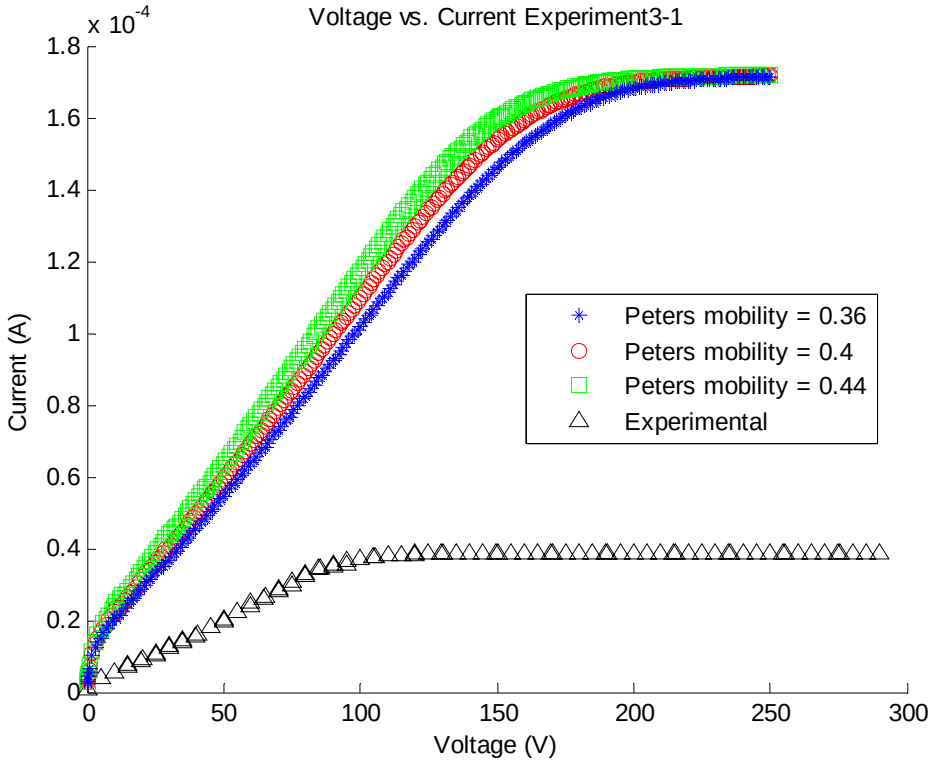

Figure E-67: Voltage vs. Current with Methane for Peters Experiment 3-1

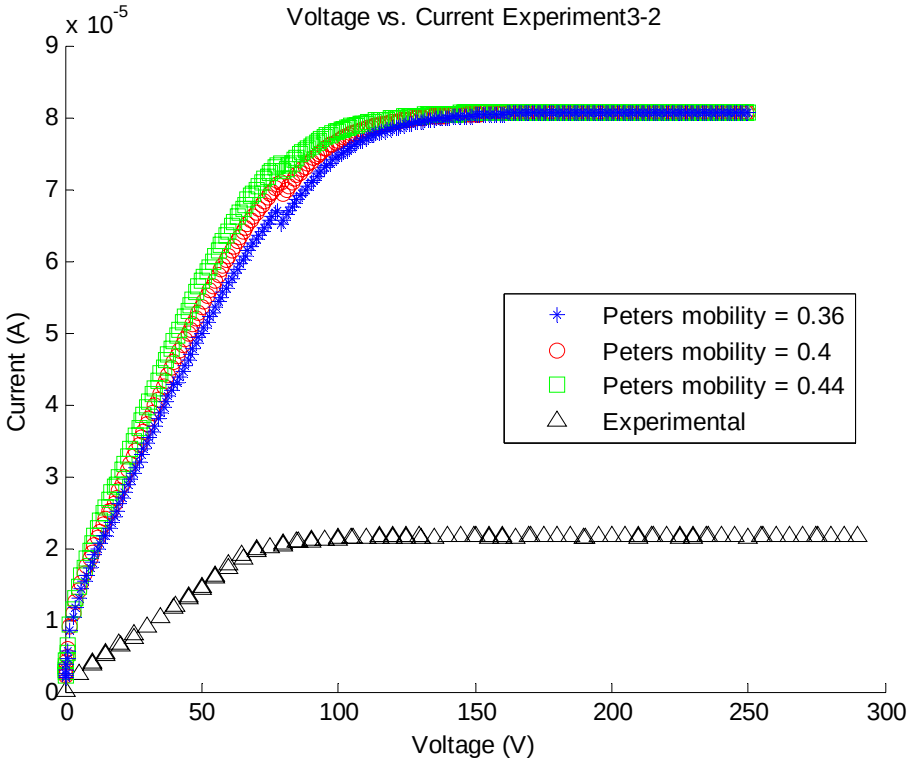

Figure E-68: Voltage vs. Current with Methane for Peters Experiment 3-2

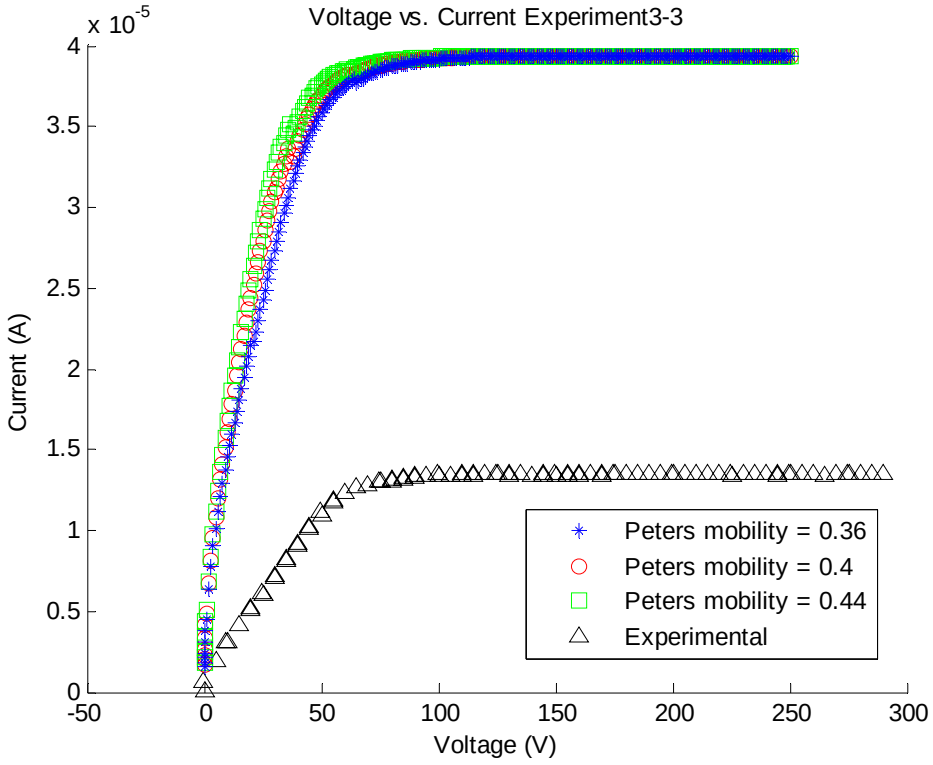

Figure E-69: Voltage vs. Current with Methane for Peters Experiment 3-3

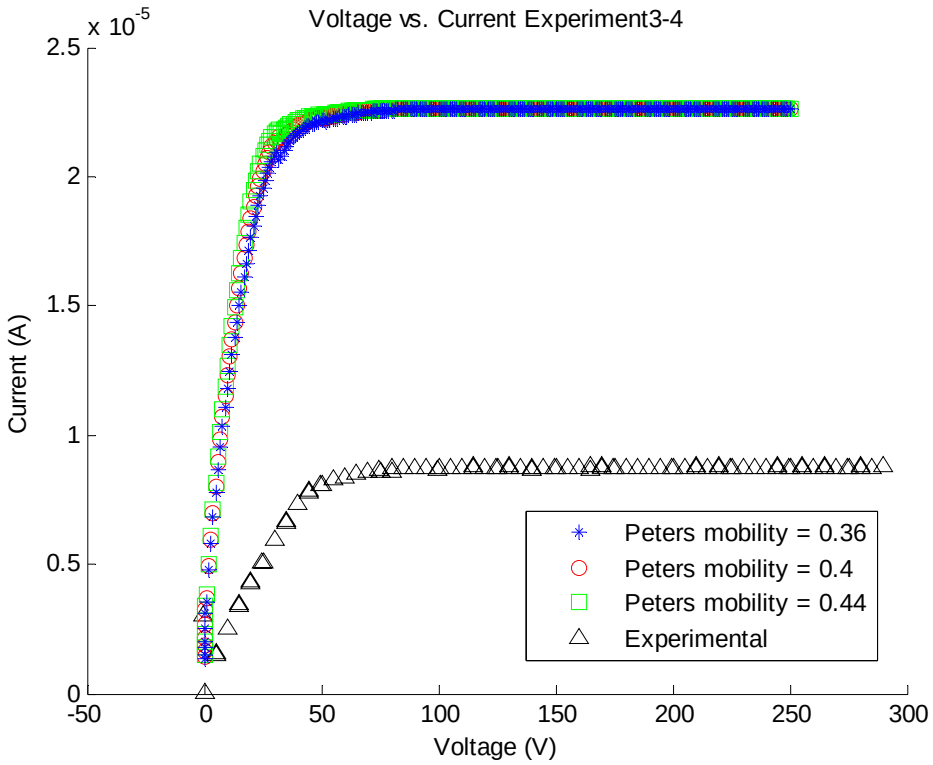

Figure E-70: Voltage vs. Current with Methane for Peters Experiment 3-4

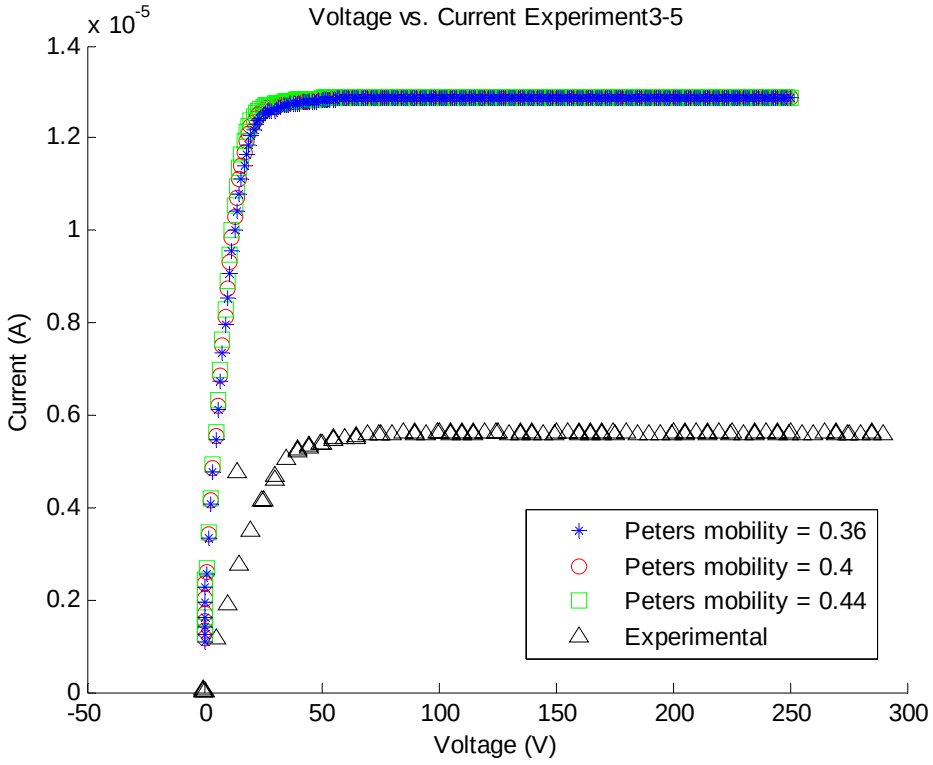

Figure E-71: Voltage vs. Current with Methane for Peters Experiment 3-5

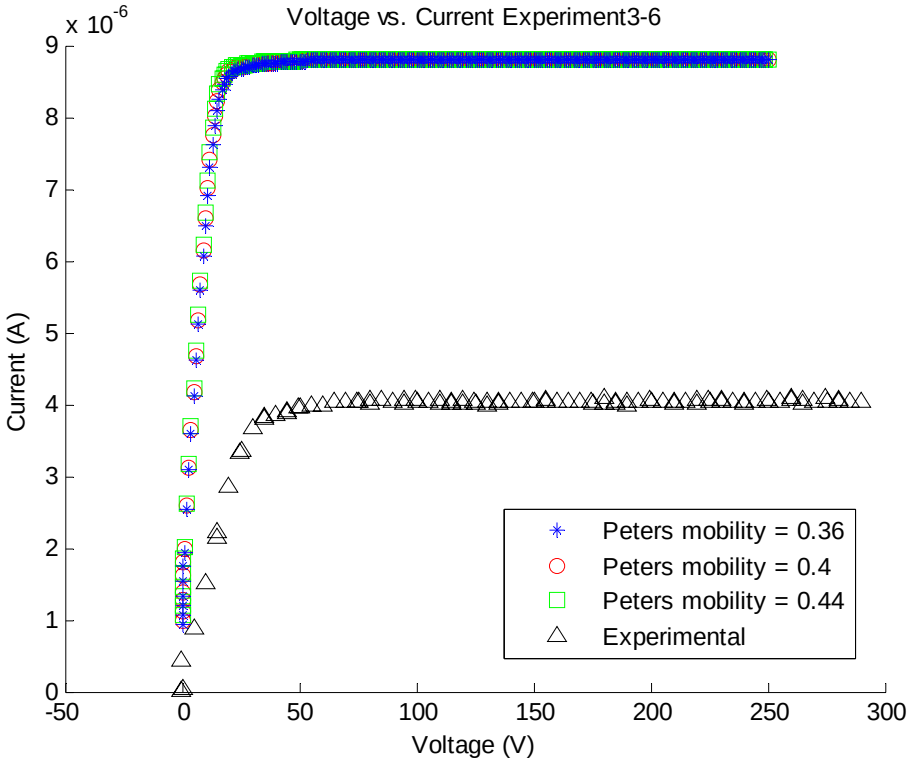

Figure E-72: Voltage vs. Current with Methane for Peters Experiment 3-6

E-5: Plots of Methane Combustion Using Four Mechanisms

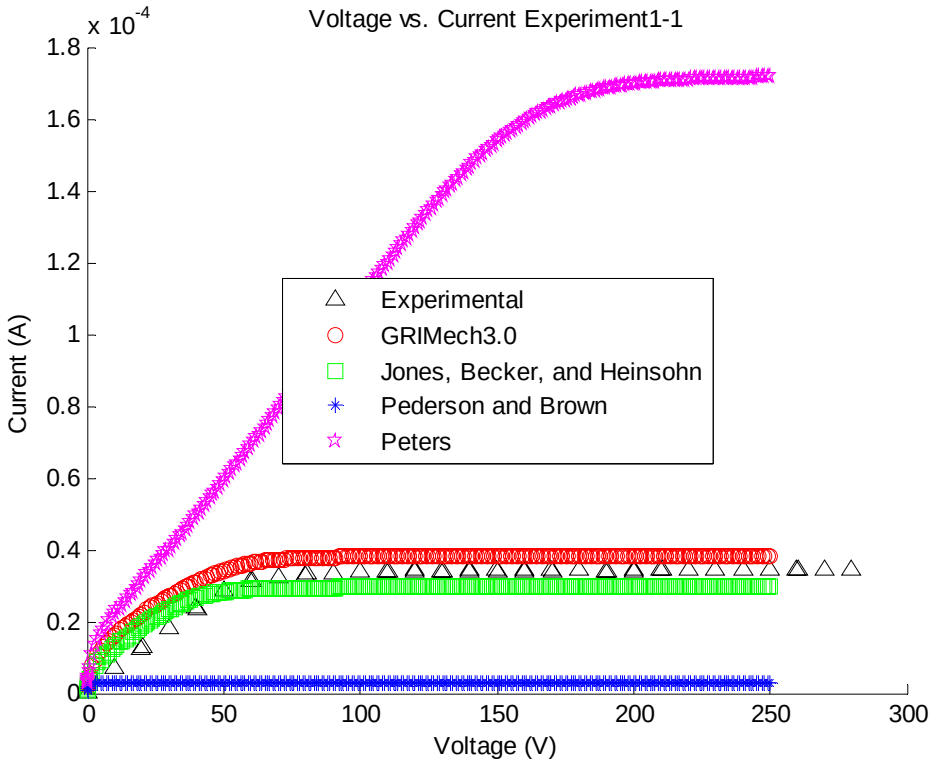

Figure E-73: Voltage vs. Current with Methane for All Mechanisms Experiment 1-1

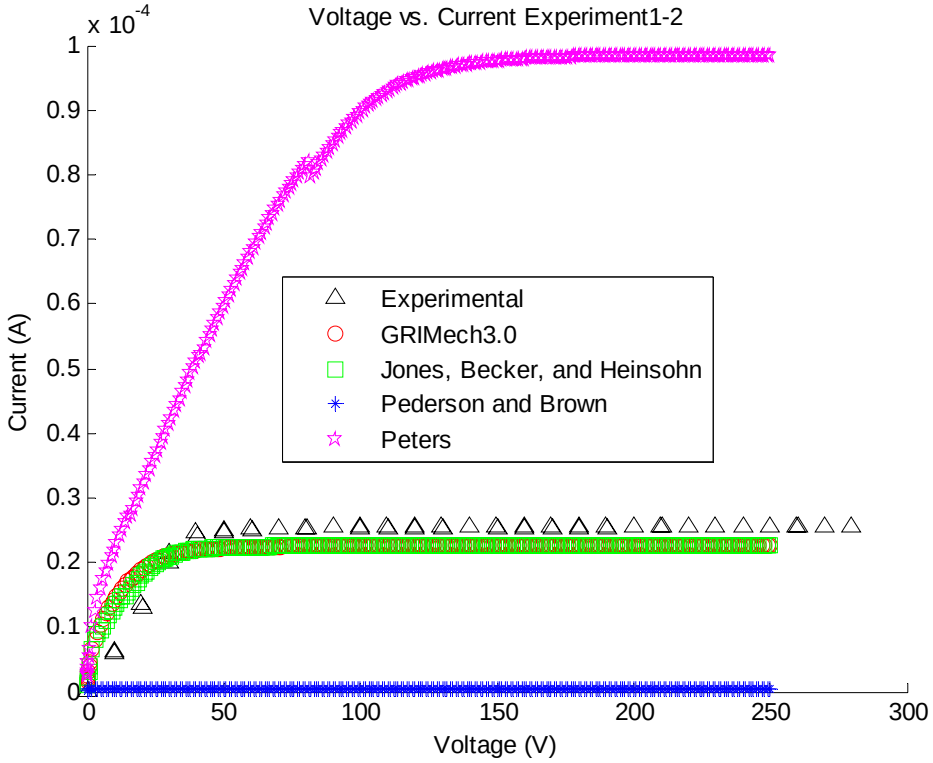

Figure E-74: Voltage vs. Current with Methane for All Mechanisms Experiment 1-2

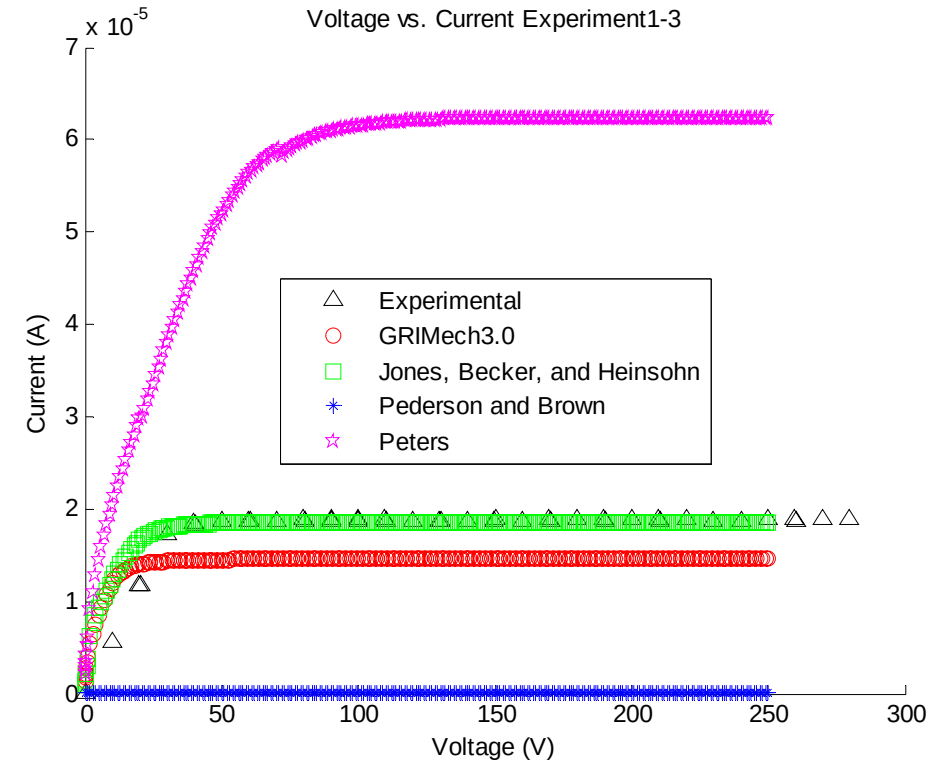

Figure E-75: Voltage vs. Current with Methane for All Mechanisms Experiment 1-3

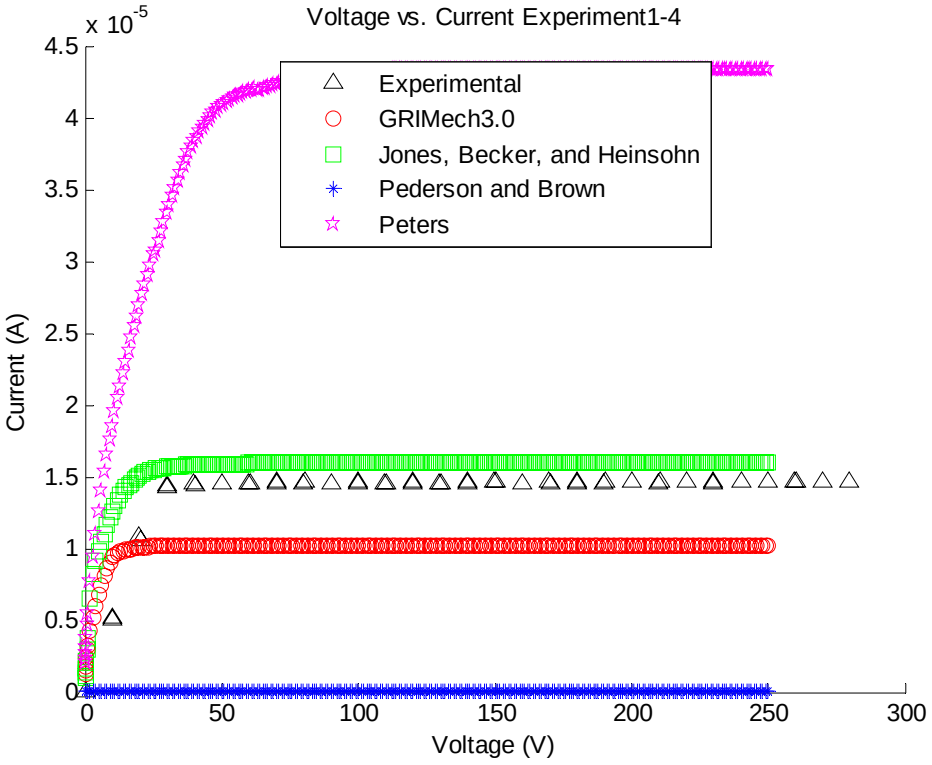

Figure E-76: Voltage vs. Current with Methane for All Mechanisms Experiment 1-4

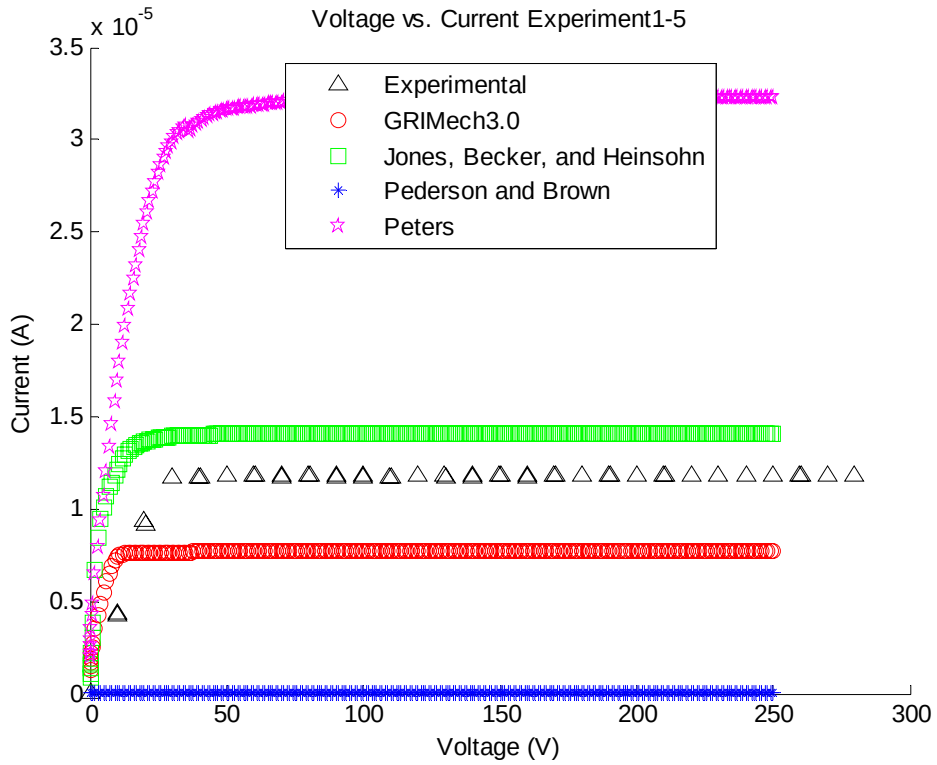

Figure E-77: Voltage vs. Current with Methane for All Mechanisms Experiment 1-5

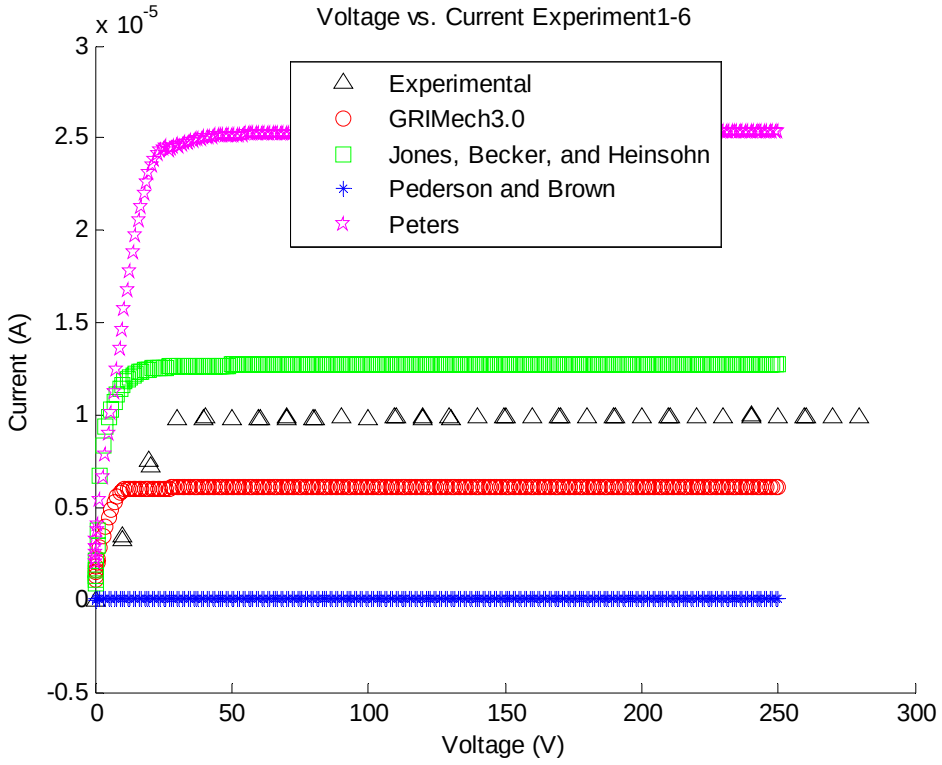

Figure E-78: Voltage vs. Current with Methane for All Mechanisms Experiment 1-6

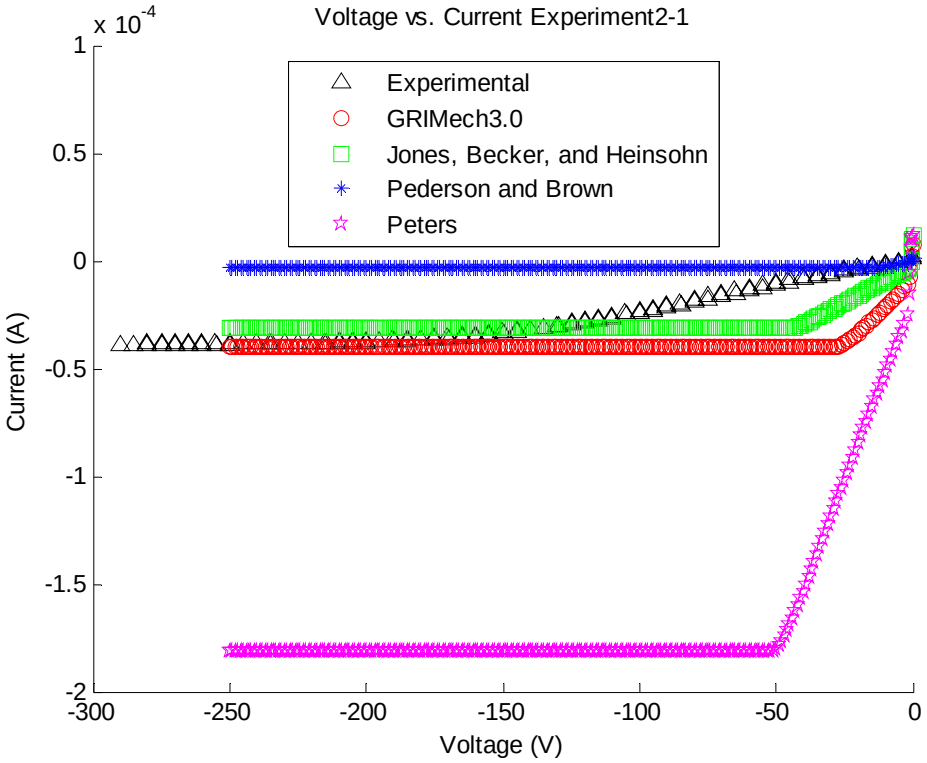

Figure E-79: Voltage vs. Current with Methane for All Mechanisms Experiment 2-1

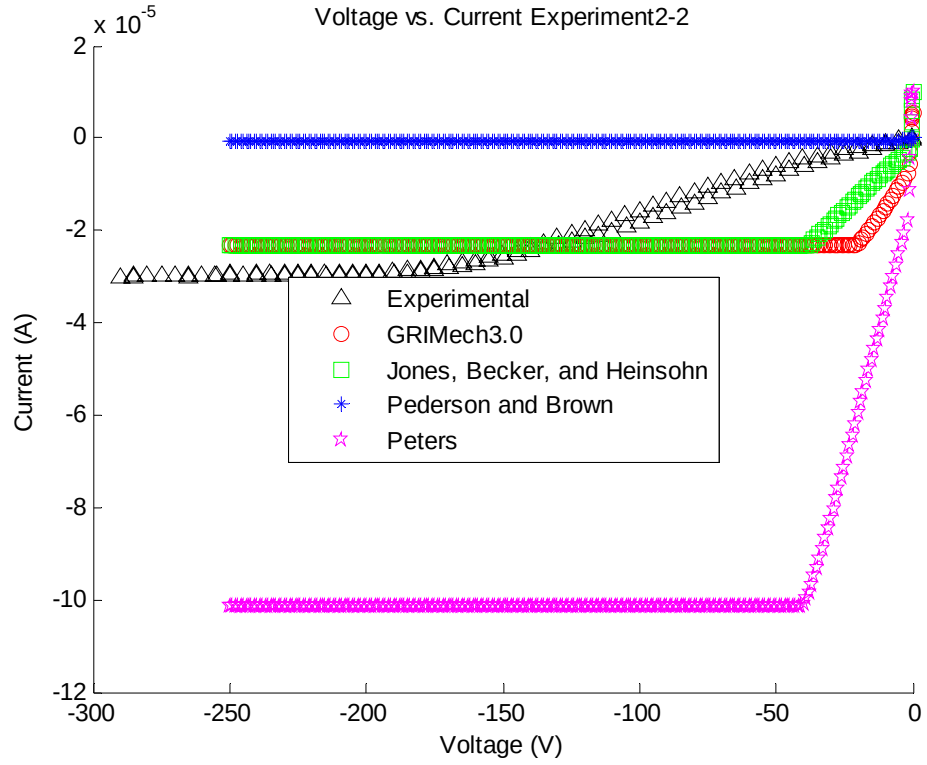

Figure E-80: Voltage vs. Current with Methane for All Mechanisms Experiment 2-2

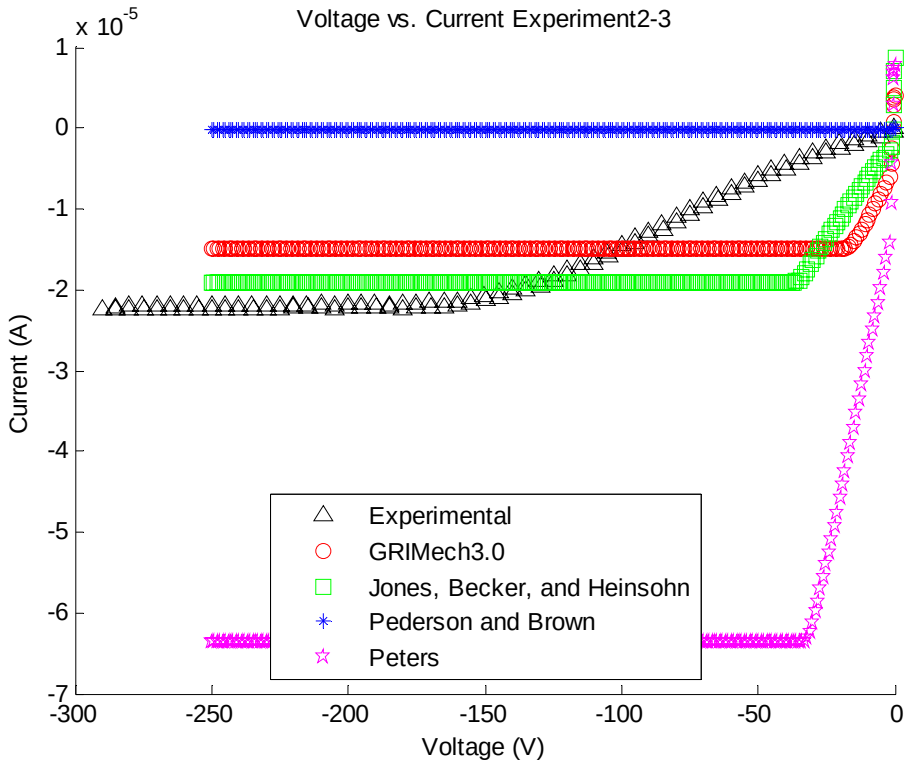

Figure E-81: Voltage vs. Current with Methane for All Mechanisms Experiment 2-3

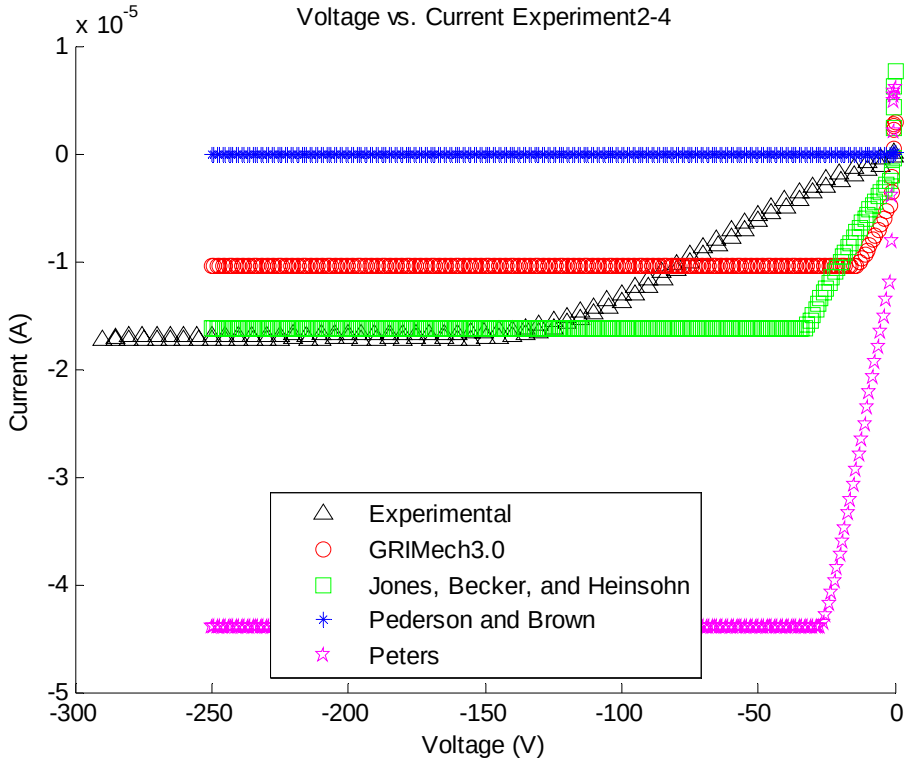

Figure E-82: Voltage vs. Current with Methane for All Mechanisms Experiment 2-4

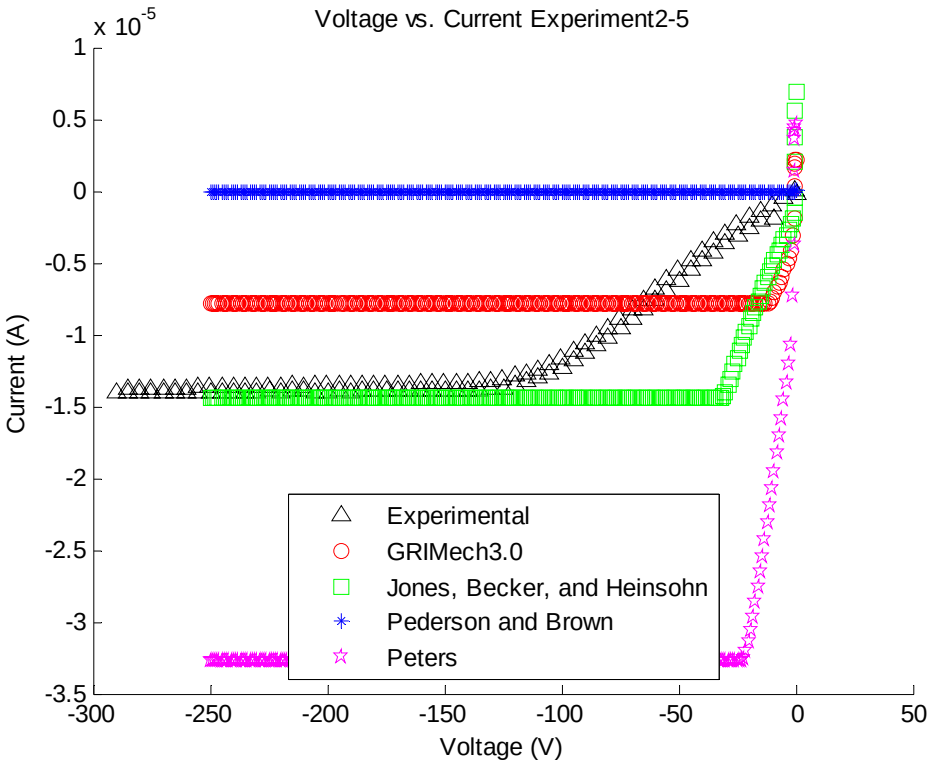

Figure E-83: Voltage vs. Current with Methane for All Mechanisms Experiment 2-5

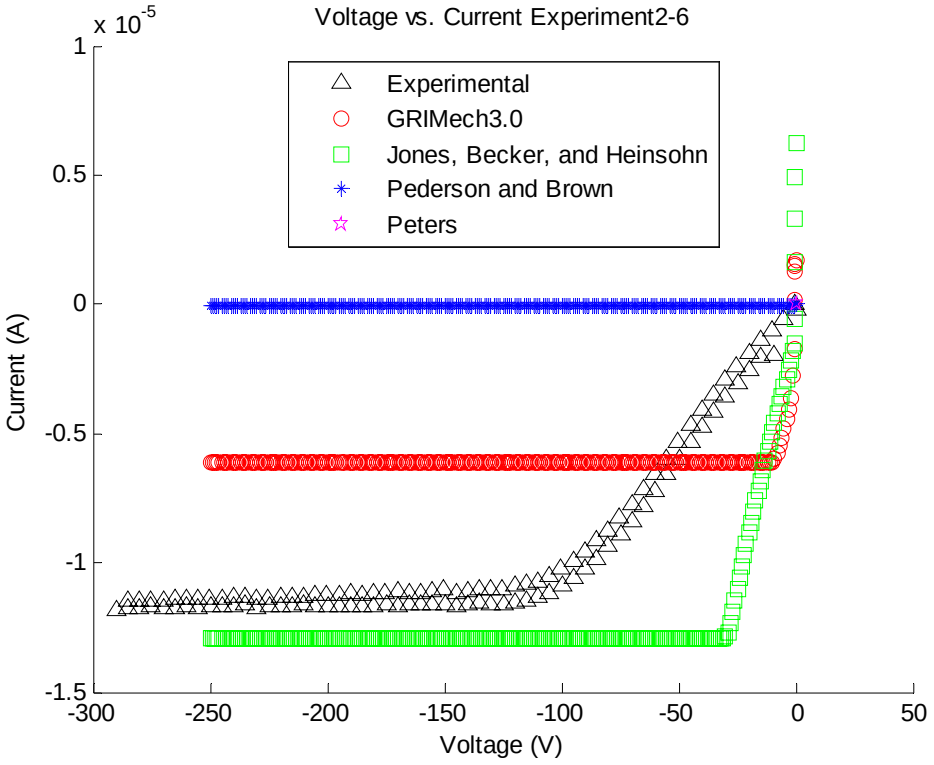

Figure E-84: Voltage vs. Current with Methane for All Mechanisms Experiment 2-6

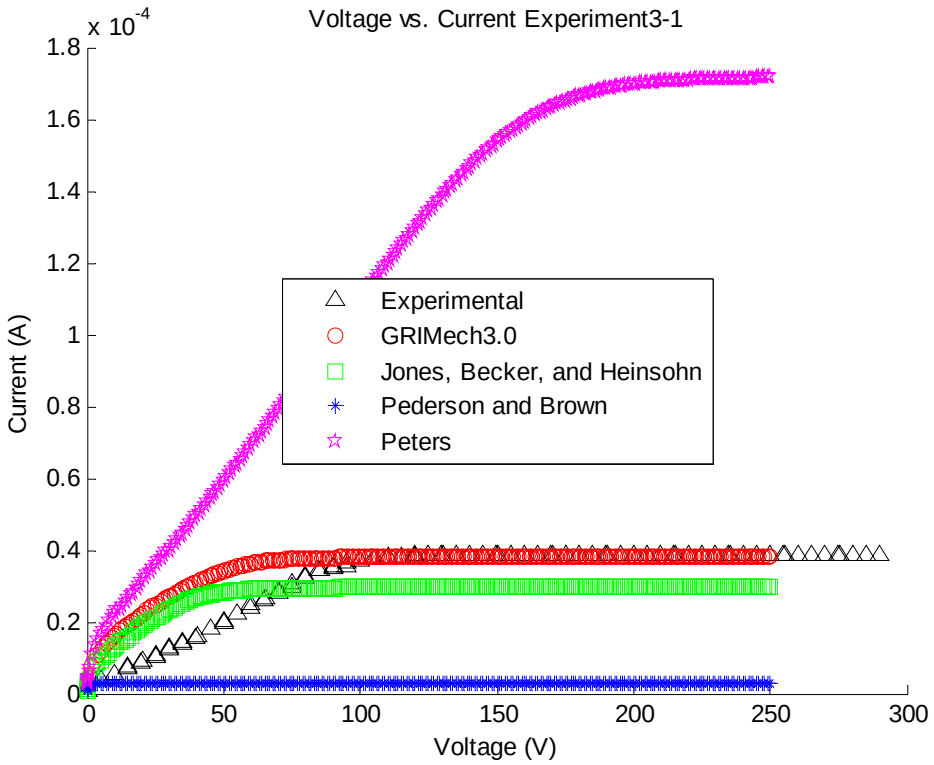

Figure E-85: Voltage vs. Current with Methane for All Mechanisms Experiment 3-1

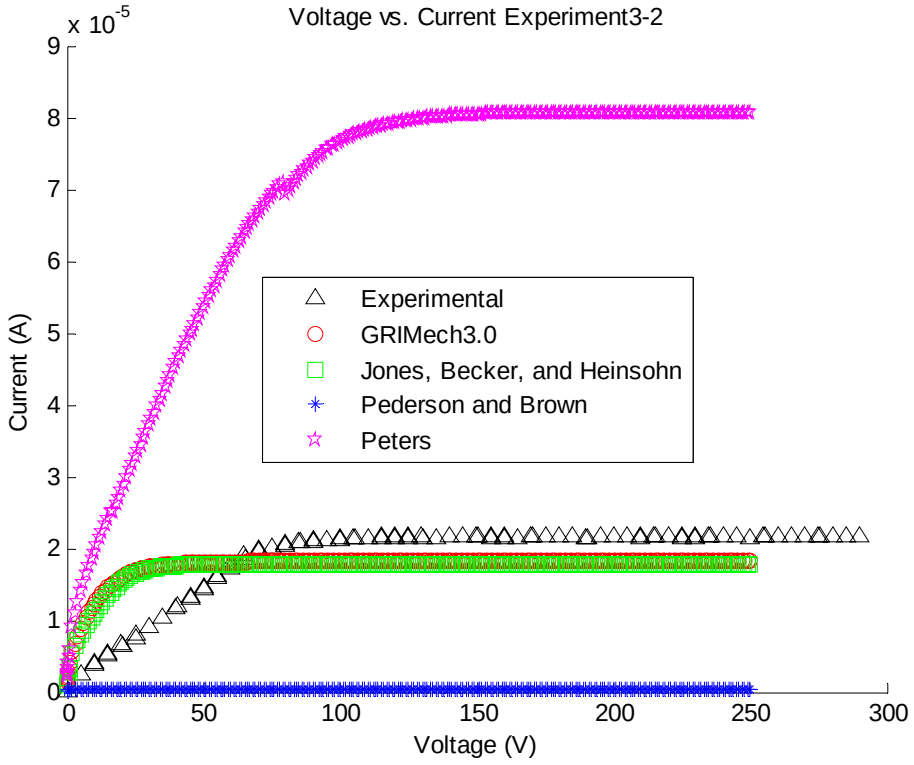

Figure E-86: Voltage vs. Current with Methane for All Mechanisms Experiment 3-2

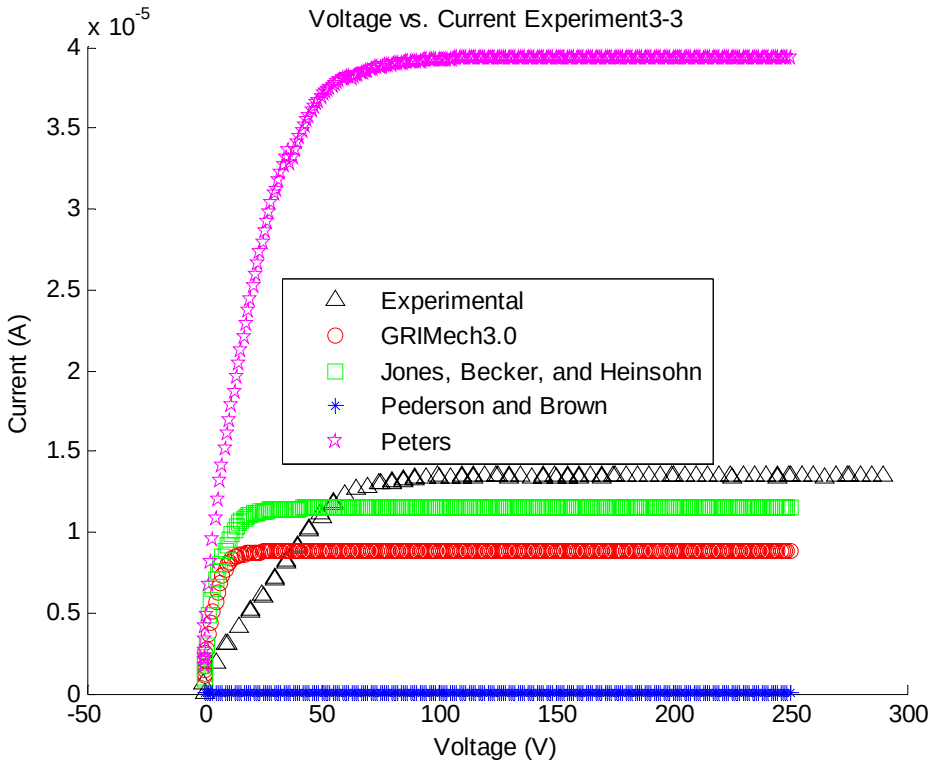

Figure E-87: Voltage vs. Current with Methane for All Mechanisms Experiment 3-3

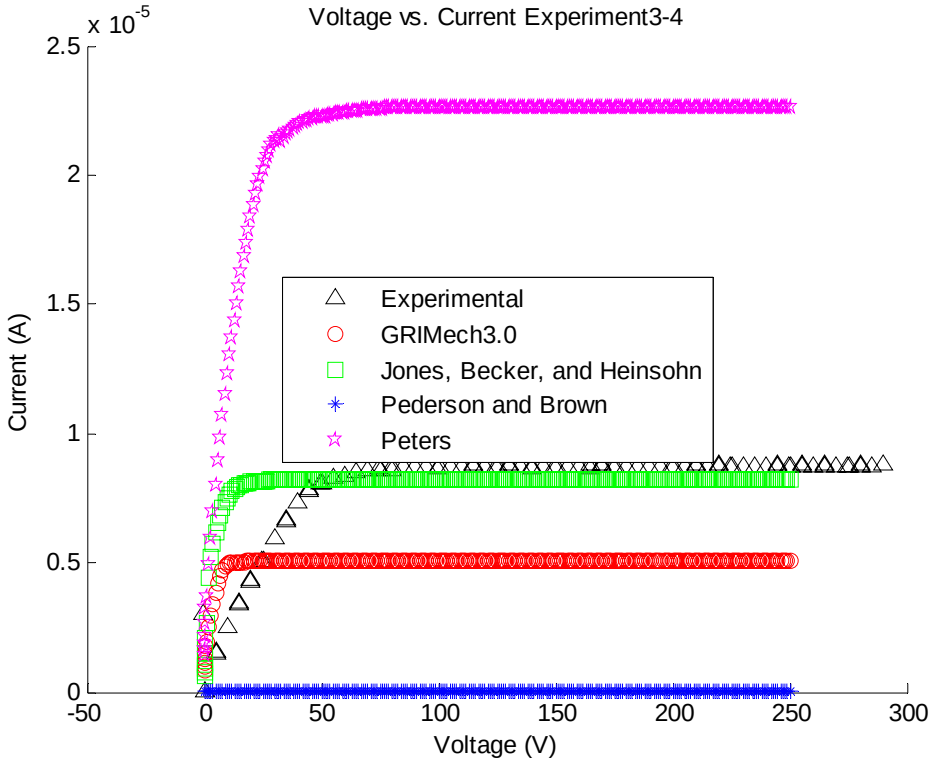

Figure E-88: Voltage vs. Current with Methane for All Mechanisms Experiment 3-4

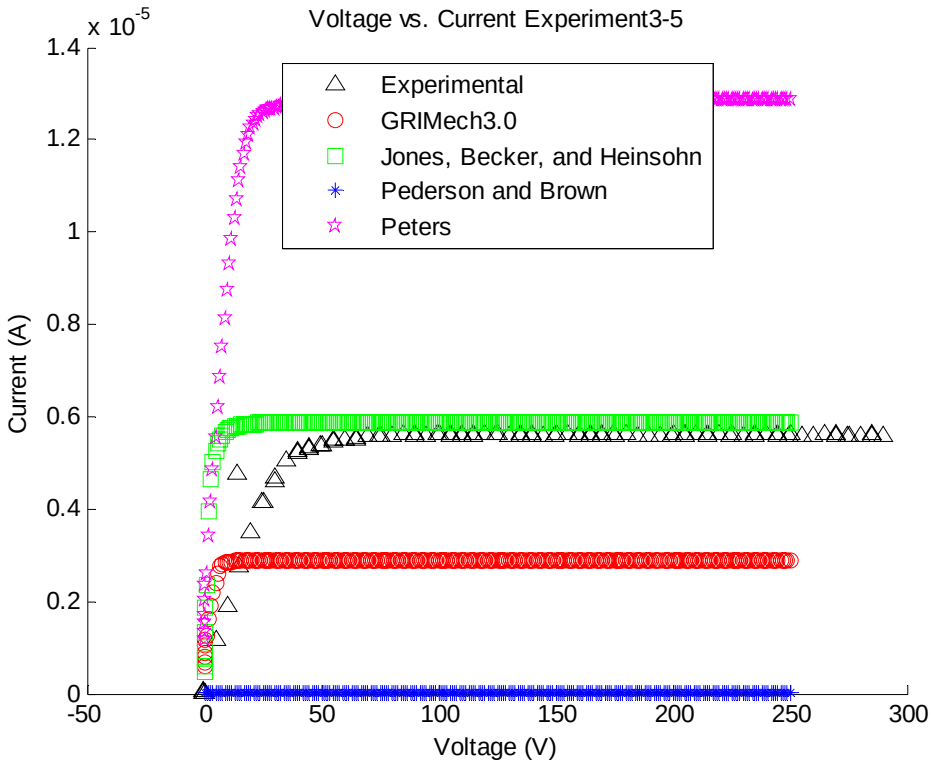

Figure E-89: Voltage vs. Current with Methane for All Mechanisms Experiment 3-5

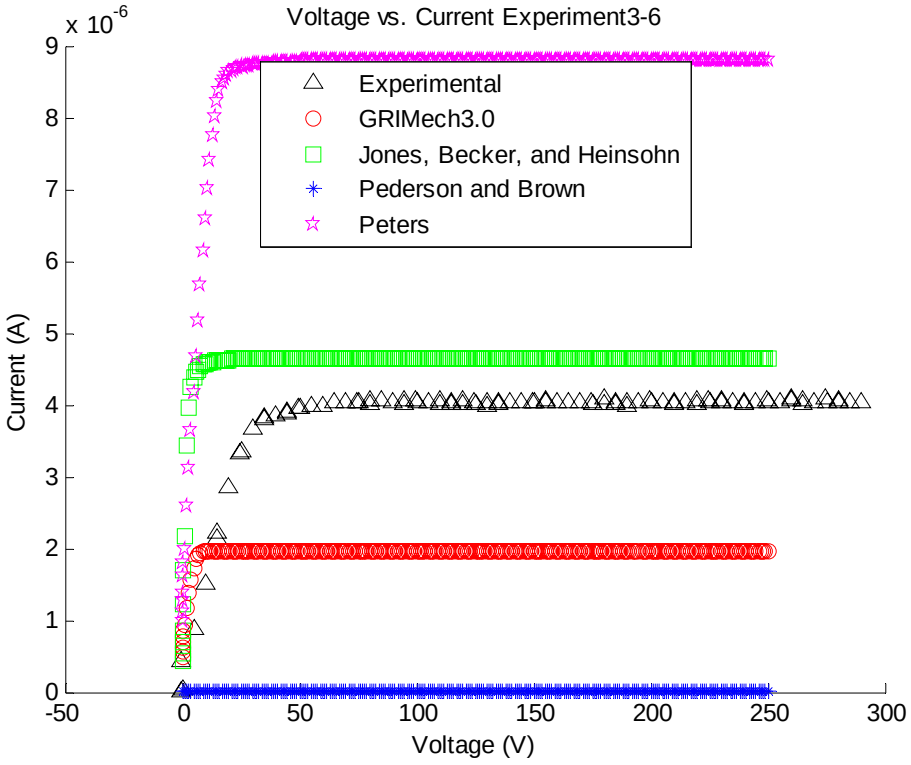

Figure E-90: Voltage vs. Current with Methane for All Mechanisms Experiment 3-6

E-6: Plots of Synthesis Gas Combustion Using Pederson and Brown

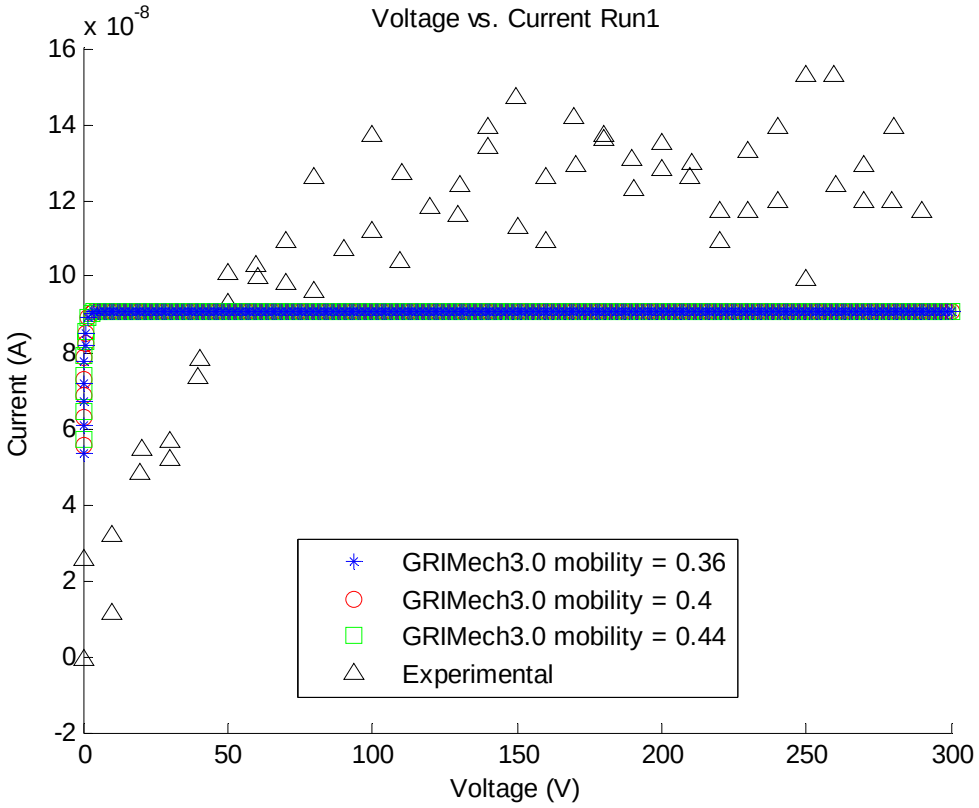

Figure E-91: Voltage vs. Current with Synthesis Gas for GRI Run 1

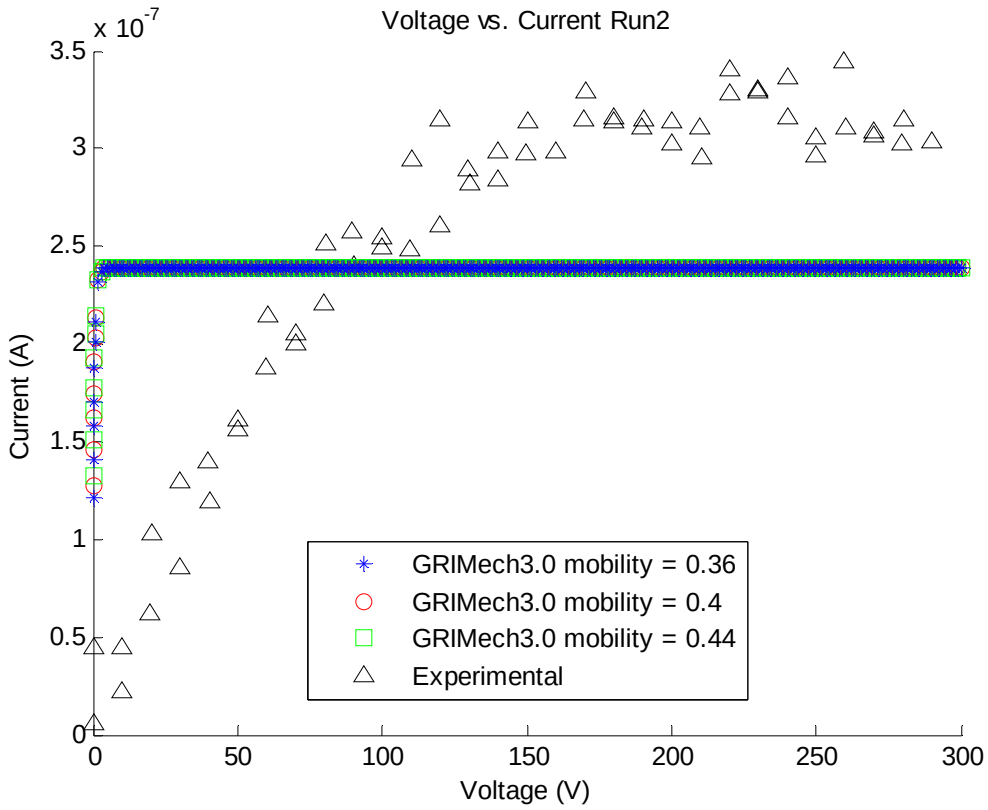

Figure E-92: Voltage vs. Current with Synthesis Gas for GRI Run 2

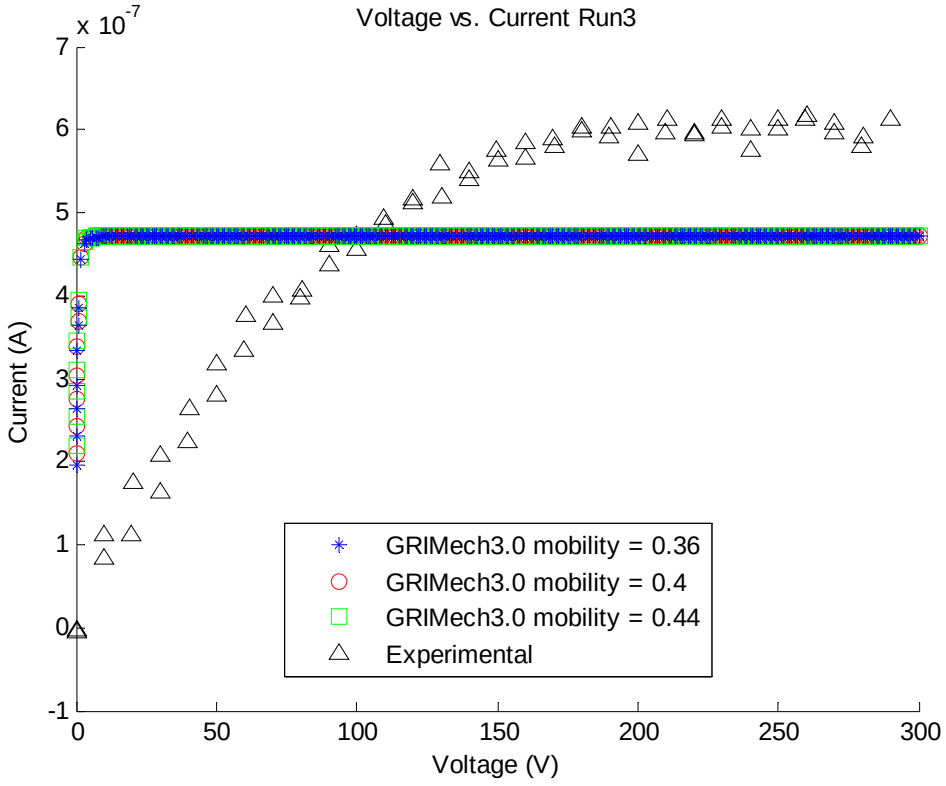

Figure E-93: Voltage vs. Current with Synthesis Gas for GRI Run 3

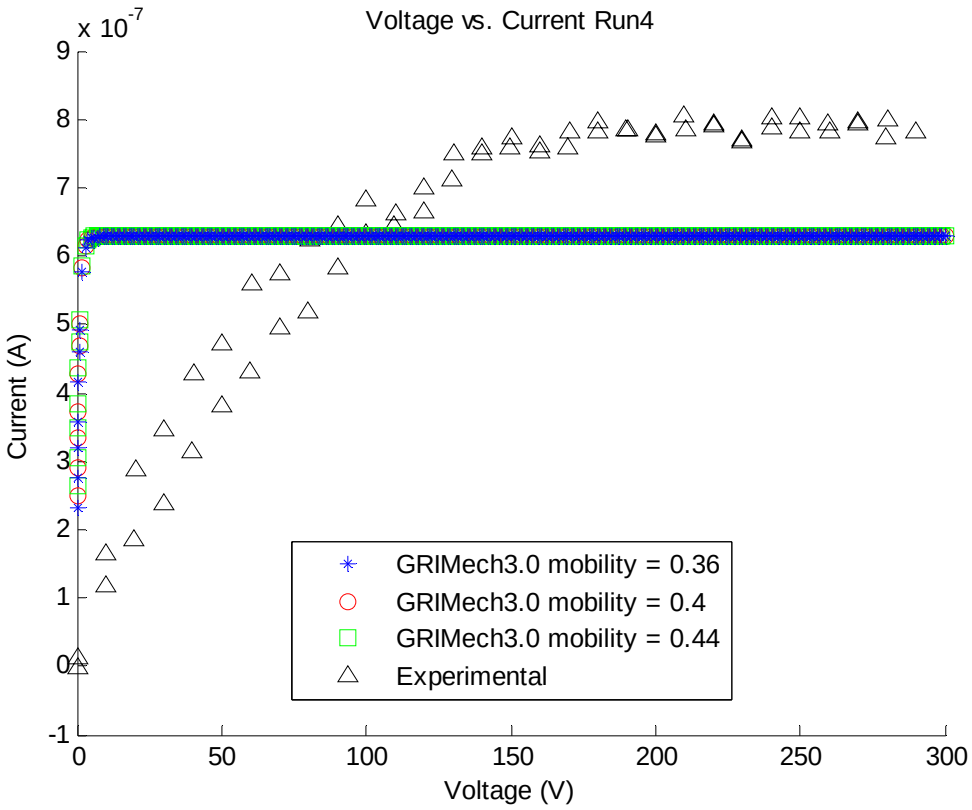

Figure E-94: Voltage vs. Current with Synthesis Gas for GRI Run 4

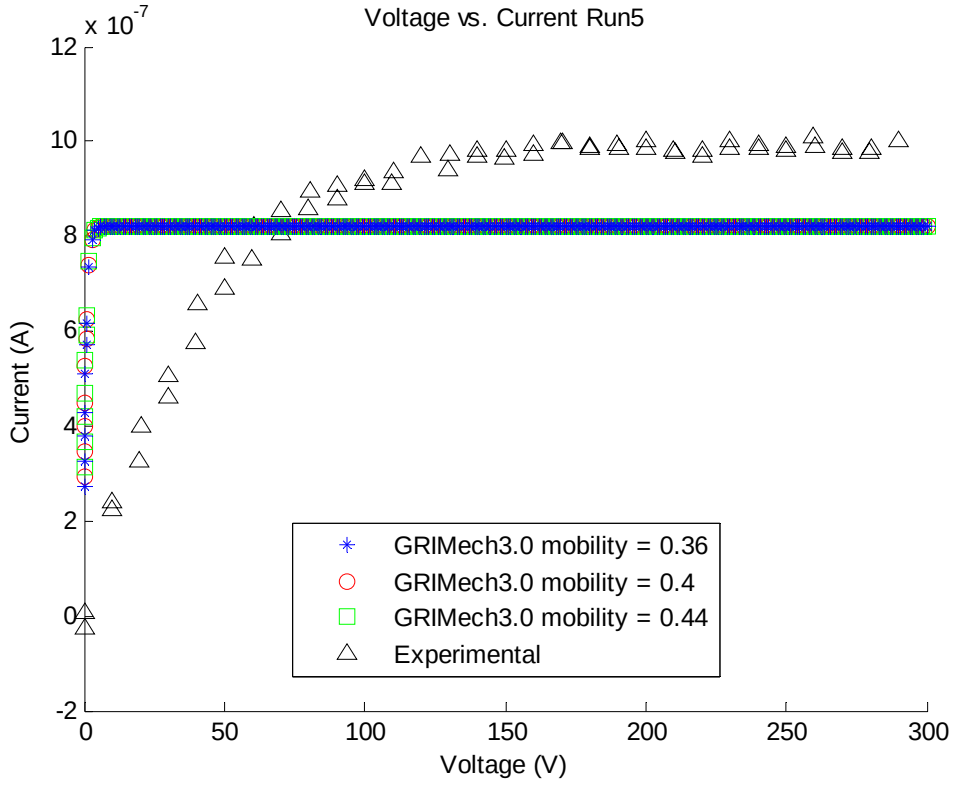

Figure E-95: Voltage vs. Current with Synthesis Gas for GRI Run 5

Bibliography

[1] Benson, Kelly A., Thornton, Jimmy D., Staub, Douglas L., Huckaby, E. David, and Richards, Geo. A., "Flame Ionization Sensor Integrated into Gas Turbine Fuel Nozzle," ASME Paper GT2003-38470, 2003.

[2] Burcat, Alexander and Ruscic, Branko, *Ideal Gas Thermodynamic Data in Polynomial Form for Combustion and Air Pollution Use*, http://garfield.chem.elte.hu/Burcat/burcat.html, 2005.

[3] Cheney, W., and Kincaid, D., *Numerical Mathematics and Computing*, 5th ed., Belmont, CA: Brooks/Cole-Thomson Learning, 2004, pp. 205-215.

[4] Eraslan, Ahmet N., and Brown, Robert C., "Chemiionization and Ion-Molecule Reactions in Fuel-Rich Acetylene Flames," *Combustion and Flame*, v. 74, pp. 19-37, 1988.

[5] Goodings, John M., Guo, Jingzhong, Hayhurst, Allan N., and Taylor, Stephen G., "Current-Voltage Characteristics in a Flame Plasma: Analysis for Positive and Negative Ions with Applications," *International Journal of Mass Spectrometry,* 2001, v. 206, pp. 137-151, 2001.

[6] Goodwin, David G., *Cantera: Object-Oriented Software for Reacting Flows*, http://www.cantera.org, 2005.

[7] Goodwin, David G., *Defining Phases and Interfaces.* http://prdownloads.sourceforge.net/cantera/definingphases.pdf?download, 2003.

[8] Goodwin, David G., *Sourceforge.net: Project Info – Cantera*, http://sourceforge.net/projects/cantera, 2005.

[9] Gordon, Sanford, and McBride, Bonnie J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications: I. Analysis", NASA Reference Publication 1311, 1994

[10] Holm, Torkil, "Aspect of the Mechanism of the Flame Ionization Detector," *Journal of Chromatography A*, v. 842, pp. 221-227, 1999.

[11] Huckaby, E. David, *Customizing Cantera's One-Dimensional Flame Code*, http://prdownloads.sourceforge.net/cantera/ModifyingLaminarRev3.pdf?download, rev. 3, 2005.

[12] Huckaby, E. David, Chorpening, Benjamin T., and Lilly, Jonathan P., Work-in-Progress Poster: "A Laminar Flame Model with Electric Field Effects," 30th Symposium on Combustion, July 2004.

[13] Jones, Fred L., Becker, Philip M., Heinsohn, Robert, J., "A Mathematical Model of the Opposed-Jet Diffusion Flame: Effect of an Electric Field on Concentration and Temperature Profiles," *Combustion and Flame*, v. 19, pp. 351-362, 1972.

[14] Kee, Robert J., Coltrin, Michael E., and Glasborg, Peter., *Chemically Reacting Flow: Theory and Practice,* Hoboken, New Jersey: John Wiley & Sons, 2003.

[15] Kee, Robert J., Dixon-Lewis, G., Warnatz, J., Coltrin, M., and Miller, J., *A Fortran Computer Code Package for the Evaluation of Gas-Phase Multicomponent Transport Properties*, Sandia National Laboratories, SAND-8246, 1986.

[16] Natarajan, J., Nandula, S., Liewen, T., and Seitzman, J., "Laminar Flame Speeds of Synthetic Gas Fuel Mixtures," *Proceedings of 2005ASME Turbo Expo – Power for Land, Sea, and Air*, Reno-Tahoe, NV, 6-9 June 2005, ASME Paper GT2005-68917.

[17] Oran, Elaine S., and Boris, Jay P., *Numerical Simulation of Reactive Flow*, New York: Elevier Science Publishing Co, Inc., 1987.

[18] Pederson, Timothy, and Brown, Robert C., "Simulation of Electric Field Effects in Premixed Methane Flames," *Combustion and Flame*, v. 94, pp. 433-448, 1993.

[19] Peters, Norbert, *Fifteen Lectures on Turbulent Combustion*, http://www.itv.rwth-aachen.de/Downloadarea/Summerschool92/SummerSchool.pdf, 1992.

[20] Python Software Foundation, *Python Programming Language*, http://www.python.org, 2005.

[21] Smith, Gregory P., Golden, David M., Frenklach, M., Moriarty, Nigel W., Eiteneer, B., Goldenberg, M, Bowman, C. T., Hanson, Ronald K., Song, S., Gardiner, Jr., William C., Lissianski, Vitali V., and Qin, Zhiwei, GRI-Mech version 3.0, http://www.me.berkeley.edu/gri_mech/, 1997.

[22] Straub, Douglas L., Thornton, Jimmy T., Chorpening, Benjamin T., and Richards, Geo. A., "In-Situ Flame Ionization Measurements in Lean Premixed Natural Gas Combustion Systems," *Western States Section – Combustion Institute Spring Meeting*, San Diego, CA, 25-26 March 2004.

[23] Streetman, Ben G., Banerjee, Sanjay, *Solid State Electronics*, 5th ed., Upper Saddle River, New Jersey: Prentice Hall, Inc., 2000.

[24] Thornton, Jimmy D., Richards, Geo. A., Robey, E., "Detecting Flashback in Premix Combustion Systems," *American Flame Research International Symposium*, Newport Beach, CA, 25-26 March 2000.

[25] Thornton, Jimmy D., Straub, Douglas L., Richards, Geo. A., Nutter, Roy S., Robey, E., "An In-Situ Monitoring Technique for Control and Diagnostics of Natural Gas Combustion Systems," *2nd Joint Meeting of the U.S. Sections of the Combustion Institute*, Oakland, CA, 25-28 March 2001.

[26] Thornton, Jimmy D., Straub, Douglas L., Chorpening, Benjamin T., Huckaby, E. David, Richards, Geo. A., and Benson, Kelly, "A Combustion Control and Diagnostics Sensor for Gas Turbines," *Proceedings of 2004 ASME/IGTI TurboExpo Meeting: TurboExpo Power for Land, Sea, and Air*, Vienna, Austria, 14-17 June 2004, ASME Paper GT2004-53392.